НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено
Завідувач кафедри
_____ С.Г. Стіренко

«___»_____2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Інженерія програмного забезпечення комп’ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Бібліотека вільного доступу»

Виконав (-ла):
студент (-ка) IV курсу, групи ІП-64 _____
Лейзьо Сергій Іванович

Керівник:
Доктор технічних наук, професор _____
Сімоненко Валерій Павлович

Консультант з нормо контролю:
Доктор технічних наук, професор _____
Сімоненко Валерій Павлович

Рецензент: _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – **«Інженерія програмного забезпечення»**

Освітньо-професійна програма **«Інженерія програмного забезпечення комп’ютерних систем»**

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____ С.Г. Стіренко

«___»_____2020 р.

ЗАВДАННЯ
на дипломний проект студенту

Лейзьо Сергію Івановичу

1. Тема проекту (роботи) *Розробка веб-додатку «Бібліотека вільного доступу» (Free Access Library)*
керівник проекту (роботи) *Сімоненко Валерій Павлович, проф., д. т. н.*,
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом проекту _____

3. Вхідні дані до проекту (роботи)
Книги, література, технічні документації інші матеріали для читання в форматі .pdf, картинки до матеріалів в форматі .jpg, інформаційні дані для відображення, доповнення та редагування бібліотеки, (логін, пароль) для входу до власного кабінету.

Вихідні дані до проекту (роботи) *матеріали в форматі .pdf, після конвертації в форматах .txt та .doc, інформаційні дані для відображення, посилання на веб-сторінки для читання онлайн.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Опис завдання; опис предметної області і напрямів дослідження; аналіз і характеристика об'єкта проектування; обґрунтування оптимального варіанта реалізації мети бакалаврської роботи; опис алгоритму і програмного забезпечення; структура проектування системи і її компонентів; основні рішення з реалізації системи в цілому і її компонентів; опис використовуваного системного програмного забезпечення; інструкція роботи користувача з системою.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)

Загальна структурна схема роботи системи, блок-схема застосованого алгоритму, функціональна схема.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітки
1.	*Затвердження теми роботи*		
2.	*Вивчення та аналіз завдання*	15.12.2019 – 03.02.2020	
3.	*Розробка архітектури та загальної структури систем*	08.02.2020 – 11.03.2020	
4.	*Розробка структур окремих підсистем*	11.03.2020 – 09.04.2020	
5.	*Програмна реалізація системи*	09.04.2020 – 30.04.2020	
6.	*Оформлення пояснювальної записки*	05.05.2020 – 10.05.2020	
7.	*Захист програмного продукту*		
8.	*Передзахист*		
9.	*Захист*		

Студент-дипломник _____

(підпис)

Керівник роботи _____

(підпис)

Анотація

В бакалаврській *дипломній* роботі реалізований веб-додаток для створення книжкової бібліотеки, реєстрацією та авторизацією нових користувачів, функціоналом що допомагає швидко та зручно використовувати запропоновані можливості.

Веб-додаток дає можливість створити бібліотеку книжок, документів, або інших видів матеріалів, керувати нею менеджером, адміністратором та надає доступ до матеріалів користувачам, яким представлений набір функцій для повноцінного використання матеріалів бібліотеки. Продукт створений на мовах програмування Java (Back-end), html та JavaScript (Front-end) у середовищі розробки IntelliJ IDEA 2019.2.3, з використанням веб-контейнеру сервлетів Apache Tomcat EE 9.0.12 та базою даних MySQL. Для візуалізації проекту використовуються інтернет-браузери.

Аннотация

В бакалаврской *дипломной* работе реализованное веб-дополнение для создания книжной библиотеки, регистрацией и авторизацией новых пользователей, функционалом что помогает быстро и удобно использовать предложенные возможности.

Веб-дополнение даёт возможность создать библиотеку книг, документов, или других видов материалов, руководить ею менеджером, администратором и предоставляет доступ к материалам пользователям, которым представленный набор функций для полноценного использования материалов библиотеки. Продукт создан на языках программирования Java (Back-end), HTML и JavaScript (Front-end) в среде разработки IntelliJ IDEA 2019.2.3, с использованием веб-контейнера сервлетов Apache Tomcat EE 9.0.12 и базой данных MySQL. Для визуализации проекта используются интернет-браузеры.

Annotation

The bachelor *thesis* deals with the theme of implemented a web-application for creating a book library with new users registration and authorization, some functionality that helps proposed features use quickly and comfortable.

Creating library, add new books or documents, or other types of reading material, manage it by managers, administrators, provide access to reading materials for users, all of these makes possible using this web-app. Users have a different functions for using library. The product was create in Java (Back-end), HTML and JavaScript (Front-end) programming languages in the development environment IntelliJ IDEA 2019.2.3, with using Apache Tomcat EE 9.0.12 web servlets container and MySQL data base.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	*Формат*	*Позначення*	*Найменування*	*Кіль-кість лис-*	*№ екзем-пляра*	*Примітка*
1.	А4		Завдання дипломного проекту	2		
2.	А4	ДП 6412. 00.000 ВП	Відомість проекту	1		
3.	А4	ДП 6412. 01.000 ТЗ	Технічне завдання	4		
4.	А4	ДП 6412. 02.000 ПЗ	Пояснювальна записка	55		
5.	А3	ДП 6412. 03.000 Д1	Функціональна схема	1		
6.	А3	ДП 6412. 04.000 Д2	Структурна схема	1		
7.	А3	ДП 6412. 05.000 Д3	Алгоритм дій користувача	1		
8.	А4	ДП 6412. 06.000 Б1	Лістинг програми	20		

				ДП 6412. 00.000 ВП		
	ПІБ	Підпис	Дата			
Розробн.	Лейзьо С.І.			Відомість дипломного проекту	Лист	Листів
Керівн.	Сімоненко В.П.				1	1
Консульт.	Сімоненко В.П.				КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІП-64	
Н/контр.	Сімоненко В.П.					
Зав.каф.	Стіренко С.Г.					

Технічне завдання
до дипломного проекту
на тему: «Бібліотека вільного доступу»

Київ – 2020 року

ЗМІСТ

					ДП 6412. 01.000 ТЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Лейзьо С. І.			Бібліотека вільного доступу	Літ.	Арк. 1 / Аркушів 4
Перевір.		Сімоненко В. П.					
Реценз.					Технічне завдання	КПІ ім. Ігоря Сікорського, Каф. ОТ, ІП-64	
Н. Контр.		Сімоненко В. П.					
Затверд.		Стіренко С. Г.					

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ВИКОРИСТАННЯ

Дане технічне завдання по темі «Бібліотека вільного доступу», розповсюджується на створення веб-додатку, який буде містити завантажені (менеджером) матеріали і надавати доступ до них зареєстрованим користувачам ресурсу. Область використання: альтернатива сучасним бібліотекам, ресурс збереження інформації для фірм, університетів, шкіл та коледжів.

2. ПІДГРУНТЯ ДЛЯ РОЗРОБКИ

Підґрунтям для розробки є завдання на виконання роботи кваліфікаційно-навчального рівня «бакалавр програмної інженерії», затверджено кафедрою спеціалізованих комп'ютерних систем Національного технічного Університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОЗРОБКИ

Ціллю даного проекту є розробка веб-додатку для збереження і відтворення матеріалів, які доступні в базі додатку, та можливістю мати свій власний кабінет користувача, щоб мати доступ до бібліотеки по темі «Бібліотека вільного доступу».

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література по мовам Java, XTHML, HTML, JavaScript, SQL. Публікації в Інтернеті, форуми, статті, документації.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги продукту розробки

- Безпека в використанні програмного забезпечення (ролі для користувачів системи з відповідним доступом).
- Сучасна платформа і технологія розробки.
- Контроль вхідної інформації і вивід достовірної бази доступних книжок (матеріалів).

5.2. Вимоги програмного забезпечення

- Операційна система Windows 7, Windows 10.
- IntelliJ IDEA 2019.2.3 і вище
- Google Chrome 81.0 і вище
- JDK 8.0 і вище
- MySQL Connector J 8.0 і вище

5.3. Вимоги апаратної частини

- Комп’ютер на базі процесора Intel Core I-3 і вище
- Оперативної пам’яті не менше 1 ГБ
- Вільного простору на жорсткому диску не менше 1,5 ГБ
- Підключення до Інтернету, веб-браузер, сервер
-

6. ЕТАПИ РОЗРОБКИ

Етап	Дата
1. Вивчення літератури за тематикою проекту	15.12.2019
2. Складання і узгодження технічного завдання	03.02.2020
3. Аналіз структури бібліотеки вільного доступу	08.02.2020
4. Створення модулів бібліотеки	11.03.2020
5. Тестування доступу до ресурсів бібліотеки через систему авторизації	09.04.2020
6. Відкладання та виправлення помилок	26.04.2020
7. Оформлення документації дипломної роботи	05.05.2020

Пояснювальна записка
до дипломного проекту
на тему: «Бібліотека вільного доступу»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ

БД – база даних

MVC – Model-View-Controller

ВСТУП

На сьогоднішній день існує велика кількість книжок, літератури, документації, збірників, журналів і інших читабельних матеріалів, що допомагають людству зберігати важливі знання, рішення, домовленості на паперах і передавати з покоління в покоління. Величезна кількість таких матеріалів потребує впорядкованості для зручності пошуку потрібних видань за назвою, автором, жанром і іншими критеріями. На протязі десятиліть створювались цілі будівлі з величезними стелями для накопичення і збереження книг. Людство надає великої цінності цьому, адже без цього неможливо було б передавати інформацію поколінням, які слідуватимуть за нами. Так само, як для нас хтось відкрив математику, біологію і фізику, і передав нам знання через написані від руки тексти, так і ми прагнемо до можливості зафіксувати свої знання. У зв'язку з розвитком технологій, для збереження великої кількості рукописних матеріалів (або надрукованих) достатньо лише одного комп'ютера, або сервера, який помістить всю інформацію на компактний і невеликий за розміром жорсткий диск. І багато людей, а особливо тих, хто володіють бібліотеками, прагнуть розробити свою власну електронну бібліотеку, за допомогою якої, кожен бажаючий зможе увійти до їхнього ресурсу без консультантської допомоги. Важливим рішенням такої проблеми є комп'ютерна програма, за допомогою інтерфейсу якої, буде виконуватись роль бібліотекара і допоможе в пошуку з конкретним результатом. Щоб покращити доступність і актуальність бібліотеки в нашому столітті потрібно створити доступну і легку у використанні базу накопичення знань. Саме вирішення цієї проблеми є завдання бакалаврського проекту. Було б дуже актуально і корисно, щоб кожен університет, школа, коледж, а також підприємство мали змогу викладати свої книжки, статті (оформлені в .pdf), наукові матеріали, потрібні для студентів, школярів і

працівників у власну базу даних, до якої кожен мав би доступ через власний кабінет. При розробці програмного веб-додатку, було розглянуто існуючі онлайн-бібліотеки і проаналізовано їх недоліки. Одними з таких були: велика кількість реклами; неможливість швидко, якісно (за декількома параметрами) знайти бажаний результат; отримати відразу потрібний формат, не використовуючи інші онлайн-сервіси; більшість книг, навіть тих, що мають бути доступними для всіх є платні, що підводить до сумніву використовуваний ресурс. А також, недоліком є те, що їхніми адміністраторами є певна група людей. Ми, в свою чергу, не розробляємо сервіс тільки для себе. Він повинен бути доступним для всіх, щоб кожен університет, школа і підприємство могли створити свій сайт з готовою базою даних і мати можливість бути адміністратором.

РОЗДІЛ 1
ОБГРУНТУВАННЯ НАПРАВЛЕННЯ РОЗРОБКИ

1.1 Опис завдання

Задання дипломного проекту - розробити веб-додаток «Бібліотека вільного доступу» (Free Access Library) в якому будуть реалізовані можливості:

1. ***Введення***: книг, літератури, технічних документацій інших матеріалів в форматі .pdf для читання онлайн, конвертації та завантаження; картинок для відображення матеріалів бібліотеки в форматі .jpg; інформаційних даних для відображення, доповнення та редагування бібліотеки; (логіну, пароля) для входу до власного кабінету.

2. ***Виведення***: в форматі .pdf; після конвертації, в форматах .txt та .doc; інформаційних даних для відображення; посилань на веб-сторінки для читання онлайн.

Окрім цього, веб-додаток має контролювати інформацію що вводиться, діагностувати помилки, реалізувати методи їх вирішення.

Підібрати ефективні методи та сучасні технології для вирішення поставленого завдання, організувати пошук, представлення та ефективне збереження інформації, налагодити систему безпеки для користувачів веб-додатку.

Навести графічний матеріал.

Створити інструкцію для роботи користувача з веб-додатком.

1.2 Опис предметної області і напрямів дослідження

Починаючи з пошукових систем, які дозволяють користувачам вводити ключові слова і знаходити потрібні веб-сайти, інформацію, веб-додатки зайняли основну роль в нашому житті. Тільки відкривши браузер, перед нами з’являться поле

вводу символів відомого нам сервісу Google, або іншої пошукової системи. Без них неможливо було б представити сучасний Інтернет, пошук сайтів. Такі сайти, як «Rozetka», «Amazon», дають можливість здійснювати покупки не виходячи з дому, що економить час, ресурси і дає можливість великого вибору товарів, сидячи при цьому в зручному кріслі, або лежачи на дивані. Одним кліком миші, або натисканням на сенсорну клавіатуру, чи голосовим вводом можна замовити ряд товарів з доставкою додому в один і той же день. Їх масштаб неймовірний. В технічній підтримці користувачів працюють десятки тисяч людей, щоб зробити наш обмін інформацією зручнішим, простішим і реальнішим.

Складно представити як раніше жили і спілкувались листами, поштою, адже, сьогодні є купа соціальних мереж, що поєднують людей зі спільники думками і бажаннями. Всього за декілька років, соціальні мережі стали настільки популярними, що є основними джерелами розповсюдження фотографій, статей, думок, відео.

Веб-додатки не закінчуються на сферах пошуку і обміну інформацією. Вони також створюються для систем, керованих за допомогою Інтернету. Все більш важливим стають Інтернет речі. Вони поєднують наші фізичні дії з віртуальними функціями, які керують системами. Нові будинки оснащуються новітніми технологіями, які заміняються повсякденні дії людини та покращують умови життя. І все це керується через веб-додаток у смартфоні, або веб-ресурс з власним кабінетом. Сьогоднішні офіси підприємств і фірм оснащені швидким Інтернет-доступом, що дає можливість моментально знаходити інформацію. Веб-конференції з різних куточків планети є звичайним явищем для нас. Крім того, програми виконують і ряд інших задач, які пов’язані з обробкою, сортуванням, відправкою повідомлень, завантаженням та редагуванням інформації. Обробка комп’ютером об’ємної бази даних дає можливість економити в часі для

знаходження книги, чи то документа в бібліотеці з 1 млн таких екземплярів.

Використання CD-дисків, чи інших носіїв інформації, скорочується тому, що передача контенту через телевізори, комп'ютери, смартфони стає все популярнішим. Хмарні сховища заміняють флешки і диски. Їх об'єм все збільшується, разом з доступністю для звичайних користувачів. Потокові передачі музики, серіалів і фільмів заміняють потребу їх зберігати на власних носіях інформації. Потрібен лиш доступ до Інтернету і вся інформація стає доступною.

Пройшли дні коли потрібно чекати на пошту від знайомого місяці і роки. За допомогою електронної пошти це виконується короткочасно, а месенджерів – в режимі реального часу. Можна без труднощів створити нове повідомлення, редагувати його, вибрати кількість отримувачів і відправити. Причому це можуть робити одночасно мільйони людей.

Смартфони стали важливою частиною 21-го століття. Вони, в якійсь мірі, заміняють комп'ютер. Ми використовуємо їх для різноманітних задач.

Банки, які оперують коштами громадян, також створили веб-додатки, за допомогою яких, можна швидко оплатити онлайн-покупку, перевести рахунок, поповнити мобільний, здійснювати регулярні платежі.

Технології можуть втрачати актуальність, але бібліотека завжди залишається цінним ресурсом для нашого суспільства. Вона допомагає знайти шукану істину. Наряду з пошуковими ресурсами, соціальними мережами та відео-переглядом, бібліотека пропонує не тільки знання, але і розвиток, відпочинок, критичне мислення, які нам необхідні при вирішенні колективних питань, сімейних, професійних. Загалом дає знання для бачення нашої поведінки в тій чи іншій ситуації. Навіть в самих недоступних і віддалених частинах світу є бібліотеки, які розширюють кругозір і дають практичні навички. Існують навіть такі, де автори створюють свої книги і заносять їх в ресурси бібліотеки.

В світі достаток людини може відрізнятися навіть в геометричній прогресії, але місце збереження книг завжди орієнтоване для всіх хто бажає, вони не залежать від достатку і мають вільний доступ.

Більшість бібліотек оснащені комп'ютерами, які мають доступ до інтернету. Деякі навіть мають планшети і ноутбуки з зручними м'якими кріслами.

Важливу роль вони відграють і у збереженні документів, цифрових копій видань, історичних та сучасних документів. Це завдання виконують не тільки для реалізації цілей інститутських бібліотек, але і для збереження здобутків певних організацій, підприємств, спільнот.

В той час, коли Інтернет зробив багато для донесення потрібної інформації по наданим запитам, бібліотеки пропонують нам також потрібні матеріали. Місце куди люди можуть приходити, знаючи що є професійні рішення задач і приклади у сторінках книжок.

На сьогоднішні день, бібліотека - набагато більше ніж простір фізичних книжок, вона включає цифрові ресурси, веб-додатки, що дають доступ до широкого кола функцій. Дуже великий бонус і значення має можливість знайти безкоштовну копію книги, яка коштує в фізичному виданні декілька сотень валютних одиниць і ця можливість здійснюється через онлайн-кабінет. Завдяки кабінету відкривається доступ до дослідницьких інструментів, наукових матеріалів, базам даних, які можуть бути як платні, так і безкоштовні. Коли в Інтернеті багато інформації, ми не можемо бути впевненими, що дані які завантажуємо не є вірусами, тому сховище даних повинно нести відповідальність за файли і документи, щоб їхня репутація була на висоті.

Бібліотека – це місце розвитку, самовдосконалення, підвищення здібності дотепно мислити, відпочивати, читаючи полюблені книги, а також місце збереження важливої інформації, що накопичувалась віками, або тієї що потрібна для

щоденного використання в побуті, фірмі, навчальному закладі або виключно для себе. Зберігання важливих документів, пригодницьких романів, технічної документації, серійних книжок та журналів і все це може використовуватися як для однієї людини так і для всіх бажаючих користувачів, що відвідають ресурс. Таке широке застосування в абсолютно різних і максимально різноманітних сферах життя, має бути забезпечене зручним інтерфейсом, зрозумілим і доступним для кожного. Надійною серверною частиною, що буде максимально зручно і правильно вирішувати поставлені задачі та відповідати сучасним технологіям. Перед виконанням проекту було проведено аналіз існуючих онлайн-бібліотек і більшість з них не дають можливості читати повні версії книжок онлайн, їх потрібно замовляти, що ставить під сумнів сам ресурс, або змиритися з цим і витратити купу часу на пошук потрібної літератури в зручному форматі, що теж не є доступним та безпечним для завантаження, бо зазвичай завантажуються віруси та інші файли, що не відповідають параметрам пошуку. Також, в більшості немає можливості читати онлайн відразу на сайті в повному обсязі.

Висновок до розділу 1

Великі підприємства, навчальні заклади мають свою літературу (матеріали, документи), але не мають можливості зробити її доступною через єдиний ресурс для своїх користувачів, за який не потрібно сплачувати.

Судячи з наведених фактів, веб-додаток, який орієнтований для збереження матеріалів в .pdf відіграє важливу роль в навчальних, підприємницьких і інших закладах. Його завдання - полегшити можливість збереження матеріалів для адміністраторів ресурсу, впорядкувати дані, дати можливість фінансово-економічному розвитку для підтримання ресурсу, керувати базою даних доступних матеріалів і користувачів системою. А також, зробити доступний для всіх інтерфейс та надати можливість вільно використовувати ресурс всім бажаючим студентам, працівникам, науковцям і іншим людям.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ І ВИБІР АРХІТЕКТУРИ, СТРУКТУРИ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ МЕТИ БАЛАВРСЬКОЇ РОБОТИ

2.1 Аналіз і характеристика об’єкта проектування

В завданні проекту вказано, що проект містить вхідні дані формату .pdf (для книжок, документів та інших матеріалів, що можуть зберігатися в бібліотеці) та .jpg (для відтворення титульного зображення матеріалів, що присутні в бібліотеці). Також інформаційні дані, які, в свою чергу, мають заповнювати та відтворювати інформацію що надходить до веб-додатку. Проект повинен в собі містити можливість автентифікації (процесу що дозволяє перевірити наявність користувача в системі та надати доступ до неї, якщо він в ній зареєстрований, або заборонити доступ, за його відсутності в ній); авторизації (процесу що слідує за автентифікацією, тобто надає права доступу до ресурсу за наявності правильного ключа логін-пароль. Користувач (з певним логіном та коректним паролем до логіну) повинен бути зареєстрований в системі і тільки тоді отримає права доступу); реєстрації (створення нового користувача, присвоєння йому ключа (логін-пароль) та надання прав доступу до веб-ресурсу); відтворення інформації що присутня в бібліотеці; доповнення і редагування; завантажувати файли у вигляді .doc та .txt (для конвертованих) і .pdf матеріалів, що присутні в бібліотеці; читання книг онлайн.

2.2 Вибір структури

В сучасному світі існує різноманітна кількість мов програмування, що допомагають у вирішенні поставлених завдань, реалізують потреби розробників для різних платформ і операційних систем.

Для оптимального варіанту реалізації мети, потрібно розглянути насамперед веб-орієнтовані мови програмування, так як, наш веб-додаток буде запускатися на сервері і користувачам потрібен доступ до нього через Інтернет ресурси (веб-сайт). А також, реалізувати гнучку та надійну серверну частину, яка буде обробляти запити від користувачів через інтерфейс та надсилати повідомлення у відповідь. Тобто нам потрібно створити динамічний веб-додаток, який буде аналізувати запит від клієнта на стороні серверного додатку (Web-контейнера), обробляти його, формалізувати і відправляти у відповідь дані у вигляді HTML сторінки. Ці дії будуть здійснюватися через протокол HTTP.

Для цього потрібно вибрати актуальну мову програмування, яка буде мати: незалежність від платформ; можливість зміни фрейм ворку та переходу до іншого, в зв’язку з розвитком технологій програмування і створення нових, покращених, новітніх засобів; продуктивну, легку у використанні, доступну для налагодження і усунення помилок серверну частину, здатну то розширення та втілення нових ідей; бібліотеки, які допоможуть в розробці; зручне і надійне середовище розробки.

Розглянемо такі мови веб-програмування, як: Python, Java, Microsoft .NET(ASP.NET), Node.JS.

2.2.1 Python

Один з самих популярних інструментів реалізації задуманих веб-проектів. Корисний буде не тільки для обробки скриптів і даних, має сучасні фрейм ворки: Sanic – веб-сервер швидкого запуску, що реалізує новий підхід в Python 3.5; FastAPI – сучасний і високопродуктивний, один з самих доступних і швидких фрейм ворків Python; Django, який має власну систему для написання HTML-файлів, які можуть взаємодіяти з даними використовуючи код Python. За допомогою цього фрейм ворку можна взаємодіяти з базами даних, веб-сторінками, обробляти HTTP-запити, маршрутизацію, також має відкритий доступ до вихідного коду. Приклади реалізації: YouTube, Instagram, Google, Pinterest, NASA і тд.

2.2.2 Java

Java була створена для забезпечення функціонування приладів які нас оточують (кондиціонери, телефони, монітори та інші) і вирішила проблему ресурсних затрат, потім для створення гри Minecraft. І вже в наш час мова застосовується в веб-розробці, і вона не втратила свого девізу «Напиши код один раз і використовуй його всюди» [1], [2].

Java Servlets – являють собою класи, технологія використовує переваги мови, та покращує продуктивність, сервлети не залежать від платформи (виконуються в JVM), що дає можливість легко змінювати самі фрейм ворки. Бібліотека Java містить широкий вибір засобів, що сприяє повноцінній реалізації можливостей Java Servlets. Є можливість обробляти низько-рівневі запити і це надає гнучкості [3][4].

За весь час існування Java Servlets, створили допоміжні технології для використання сервлетів. Такі як JSP, JSF, Spring.

JSP – зручний механізм розробки починаючи з 1998 року, завдяки якому можна компонувати Java-код і HTML-код.

JSF – більш сучасна і складніша технологія для взаємодії інтерфейсу з серверною частиною (В 2016 вийшла оновлена версія 2.2.13). Підходить для реалізації легкого доповнення проекту, в разі необхідності. Зручно використовувати разом з доступними і високопродуктивними компонентами PrimeFaces, які можна повністю налаштувати під свої потреби, та створити інтерфейс користувача, використовуючи об'ємну бібліотеку тегів. Як і для JSF так і для PrimeFaces є велика кількість форумів та обговорюваних питань про налагодження веб-додатку, усунення помилок, з якими можна зіткнутися під час розробки програмного забезпечення. [9]

Spring Framework – один з самих популярних, багато налаштувань виконуються автоматично, можна підключати тільки потрібні модулі, що зменшує кількість залежностей та навантаження на сервер.

Для розробки програмного коду по стандартам мови програмування Java доступна велика кількість книжок [1], [2], [3], [4], [5], [7].

2.2.3 Microsoft .NET (ASP.NET)

Розробка компанії Microsoft, на даний момент, доступна для операційних систем Windows, але у фазі розробці й для інших операційних систем. Середа розробки дає можливість використовувати обмежену кількість мов, які сумісні з (.NET). Головною проблемою є те, що, на даний момент, є обмеження в багатоплатформенності.

2.2.4 Node.JS

Серверна реалізація мови програмування JavaScript (платформа). Відмінність від інших мов програмування веб-сервера в тому, що виконується не в браузері, а на стороні сервера. Перевагою є можливість одночасно виконувати декілька операцій, які між собою не пов’язані; на сервері і на клієнті використовується однакова мова програмування, що дуже спрощує розробку; гнучка та лаконічна мова. Основними недоліками є відсутність повноцінної IDE, проблеми з контролем пам’яті і налагодженням можливих помилок, які виникають в коді під час розробки програмного забезпечення.

2.2.5 PHP

Призначений для програмування на стороні веб-сервера, є дуже популярний в цій області. Більшість сайті написані саме за допомогою цієї мови. Простий синтаксис дає можливість створювати навіть великі, складні проекти і реалізовувати бажані алгоритми. Має широкий спектр бібліотек для створення проектів на найбільш використовуваних операційних системах (Windows, Linux, Mac OS). Продовжує оновлюватись і постійно виходять нові версії. Підтримує роботу з великою кількість відомих баз даних (Oracle, MySQL, PostgreSQL) та веб-серверів.

Отже, проаналізувавши великий спектр мов програмування для веб-додатків, можна зробити висновок, що для реалізації завдання бакалаврського проекту підходять не всі мови, а точніше: Python, Java, Node.JS, PHP. Зважаючи на те, що в мене є вміння і навички в розробці програмного забезпечення на мові Java більш ґрунтовні і обширі, я надаю перевагу саме цьому варіанту. Можу з впевненістю сказати, що, незалежно від видатних фрейм ворків, бібліотек, сама мова Java

являється гнучкою, надійною, безпечною, незалежною від платформ, актуальною, з відкритим вихідним кодом бібліотек та великою їх кількістю, що забезпечить максимальну зручність в створенні, реалізації, налагодженні і оформленні програмного коду для веб-додатку.

Гнучкості надає ООП. Принципи якого є: абстракція – дозволяє працювати з об'єктами, не вдаючись в особливості абстрактного класу (Наприклад: телефон абстрактний клас (смартфон наслідувач, телефон з клавішами наслідувач)), інкапсуляція (реалізацію класу зробити приватною а методи для користувача – публічними, це підвищує безпеку додатка), наслідування (запозичення методів іншого класу), поліморфізм (один інтерфейс – різна реалізація).

Також, в Java містяться технології, що дають можливість працювати з HTML-сторінками і JavaScript-кодом (JSP, JSF, Spring Framework), за базами даних та контейнерами сервлетів типу: Apache Tomcat, JBoss, GlassFish і інші, які представляють собою сервер і забезпечують життєвий цикл сервлетів. Коли питання стоїть про актуальність методів і технологій розробки слід звернути увагу на JSF і Spring Framework тому, що JSP є трішки застарілою (останнє оновлення версії 2.3 в 2013 році). Для виконання завдання потрібна програма яку можна буде легко доповнювати, розширювати можливості і в цьому нам допоможе JSF, а у разі потреби можна без особливих затрат змінити технологію на Spring Framework. Інтегровані середовища розробки (IntelliJ IDEA, NetBeans, Eclipse), які вже декілька десятків років підтримують середовище інструментів, налагодження, компілювання і редагування коду, а також, мають офіційні сайти з технічною підтримкою користувачів, надають впевненості та полегшують розробку програмного продукту.

Тепер перейдемо до бази даних для зберігання інформації. Нам потрібна гнучка проста в використанні реляційна база даних, з можливістю створювати різні

типи об’єктів, в якій дані мають логічну послідовність і зв’язок за певними критеріями (id, name, username and other). Щоб ми могли запросто взяти потрібні нам дані за допомогою запитів і так само доповнити їх, редагувати, видалити, фільтрувати, сортувати, знаходити за критеріями запиту. В цьому нам допоможуть: MySQL, PostgreSQL.

2.2.6 MySQL

Сама популярна в області серверних БД. В Інтернеті є велика кількість інформації про її налаштування, оптимізацію. Велика кількість підтримуючих типів даних. Також при установці пропонується програма для полегшення роботи з цією БД. Багатофункціональна і підтримує велику кількість SQL запитів. Швидка та надійна.

2.2.7 PostgreSQL

Сама розвинена БД. Розробники прагнуть виконувати майже всі стандарти запитів SQL. За об’єктно орієнтованої структури має можливість до розширення. Виконує складі SQL-запити, забезпечує передову цілісність даних.

Проаналізувавши передові бази даних, я зупинився на виборі MySQL, тому, що з двох ринкових гігантів потрібно вибрати щось одне і більше за все, для не самого масштабного проекту, розробкою якого займається одна людина, підійде саме MySQL, яка гарантує швидке читання даних, що навіть покращить нашу продуктивність і швидкість отримання даних. А запити в БД будуть не настільки складні, як для використання PostgreSQL.

2.3 Вибір архітектури

Архітектура програми – це логічна структура її компонентів, в якій всі компоненти взаємодіють для вирішення поставленого завдання, їх властивості і зв’язки у вигляді однієї продуктивної системи.

Архітектурні патерни проектування – це і є вирішення архітектури нашого коду, вони включають схеми наших компонентів, їх зв’язки і взаємодію одного з іншими. Вони незалежні від мови програмування, чи структури програми.

Існує декілька видів архітектурних патернів для веб-програмування. Такі, як: MVC, MVP, MVVM та інші.

MVP (Model-View-Presenter) – схожа до MVC, отримує дані від View через Presenter і не може отримувати дані відразу з View, як MVC.

MVVM – зв’язаність між Model і View проходить через ViewModel, така концепція дуже зручна для тестування веб-додатку, коли нас не потрібно змінювати Model, а потім перевіряти View, ми можемо зразу перевірити у ViewModel, перед тим, як дані будуть відображені у View.

Для вирішення проблеми проектування додатку, використовуємо архітектурну модель проектування MVC. Вона допомагає розділити програмний код на модель, представлення і контролер, що мають певну логічну роль у розділенні програмного коду інтерфейсу користувача від управляючої логіки програми. При чому, кожен компонент може існувати окремо. Модель служить для маніпуляцій з даними (у нашому випадку з базою даних), представлення – це відображення інтерфейсу користувача у вигляді сторінки HTML, контролер – для управління всіма даними (через нього проходять всі операції маніпулювання, виконання функцій, зміни інтерфейсу).

Процес виконання запиту починається за допомогою інтерфейсу користувача, контролер аналізує дані, видає помилку у разі некоректності, або направляє дані для обробки бізнес-логікою (функціями). Йому повертається результат, який він інтерпретує для представлення. Користувач отримує результат виконання запиту. Це архітектурне рішення допоможе відділити створення графічного інтерфейсу від бізнес-логіки та класів, що відповідають за отримання даних з БД. Тобто класи будуть поділені на Model, View, Controller.

До Model віднесемо класи User, UserRole, Book. User – клас який буде відповідати за користувача та містити його ім’я, пароль доступу до системи і баланс. UserRole – буде містити роль користувача в системі, яке прив’язане до імені. Book – вся інформація про книгу (автор, ціна, назва, короткий опис, ідентифікатор в базі даних, жанр, мова, рік видання, шлях місцезнаходження). А також, перерахування Format, в якому будуть необхідні формати, для зручності їх використання і легкого доступу в коді, без дописування строки символів.

Контролерами будуть класи AdminController, ManagerController, UserController.

UserController відповідатиме за управління сторінкою користувача. В ньому будуть функції для отримання імені користувача, перевірки балансу, поповнення рахунку, перевірки на можливість завантаження матеріалів доступних у бібліотеці, конвертування в текстовий формат, конвертування в формат Word-документа, функція для читання книг онлайн. А також, змінні для передачі інформації для заповнення таблиці даних про книги.

ManagerController відповідатиме за управління сторінкою контент менеджера. В ньому будуть функції для сортування книг, фільтрування, видалення. З додаванням нових книжок і редагуванням, йому будуть допомагати сервлети AddBookServlet, EditBookServlet. AddBookServlet викликатиметься при додаванні нової книги, отримуватиме дані з полів вводу та додавати їх у базу даних.

EditBookServlet буде отримувати дані для редагування присутніх у бібліотеці матеріалів, оновлювати таблицю даних.

AdminController повністю керуватиме сторінкою адміністратора, де реалізована функція видалення користувачів.

За представлення відповідатимуть сторінки користувача (user_hello.xhtml), контент-менеджера (manager_hello.xhtml) і адміністратора (admin_hello.xhtml), для яких буде створено інтерфейс взаємодії з системою, вивід даних та компоненти що допоможуть виконувати задачу веб-додатку.

Всі класи, які є управляючими контролерами будуть з анотацією @ManagedBean, яка дасть можливість їх використовувати і застосовувати у відображенні (View). Тобто прив’язати кнопки і інтерфейс до функцій контролера, які будуть реагувати на дії користувача і виконувати обробку даних на сервері.

Контролерам будуть допомагати сервлети, які мають змогу надсилати повідомлення про помилку та контролювати введені дані. Також вони допоможуть при завантаженні нових даних (файлів, картинок).

Ще однією з важливих компонентів нашого веб-додатку буде Maven. Це допоміжний інструмент для збирання Java-проекту, створення документації, створення дистрибутива. Maven незалежний від операційної системи, сторонні бібліотеки, які потрібно підключити до використання, прописуються в скрипті, що швидко їх підключає з бази даних. Є можливість контролювати версії цих бібліотек і легко оновлювати їх при такій можливості. Проект автоматично збирається на сервері, що не потребує написання лишніх команд для запуску. Компонент має чудову інтеграцію з різними середами розробки, що зробить наш проект універсальним і гнучким.

2.4 Програмне забезпечення

Інтегроване середовище розробки для реалізації програмного забезпечення – IntelliJ IDEA 2019.2.3, яка вже 20 років займає одну з перших позицій інтегрованих середовищ для розробки на Java. Компанія-розробник (JetBrains) надає безкоштовну ліцензію для студентів на використання повної версії продукту, а саме, для розробки веб-додатків, тобто платформа (Java 2 Enterprise Edition), яка доступна тільки у повній версії програмного забезпечення (IntelliJ IDEA 2019.2.3 Ultimate Edition). При створенні проекту обираємо Java SE Development Kit 11.Для використання БД MySQL, потрібно підключити до проекту MySQL Connector J 8.0. Для полегшення роботи і створення нових таблиць, редагування даних, MySQL при установці, дає можливість встановити допоміжну програму MySQL Workbench 8.0.

В якості контейнеру сервлетів ідеально підходить Apache Tomcat EE 9.0, який також потрібно підключити до проекту в налаштуваннях «Конфігурації» [8].

Висновок до розділу 2

Проаналізувавши завдання, вибрано мову програмування Java для серверної частини, HTML і JS для інтерфейсу користувача; архітектурний шаблон MVC; програмне забезпечення, яке створить комфортні умови для взаємодії технологій і розробці програмного продукту.

РОЗДІЛ 3
ОПИС РОЗРОБКИ

3.1 Опис алгоритмів

3.1.1 Алгоритм реєстрації

Для того, щоб почати користуватися ресурсом «Бібліотека вільного доступу» (Free Access Library), необхідно створити особисту сторінку (кабінет), в якому буде міститися інформація про суму коштів користувача, його ім’я. Скориставшись формою реєстрації, до якої є посилання з головної сторінки, достатньо вказати логін, пароль, та підтвердження пароля. Якщо майбутній користувач ввів всі дані правильно, то виконується пошук користувача (з введеним логіном) в базі даних всіх користувачів системи. Якщо такий користувач з таким логіном вже існує, то під час реєстрації буде переправлено на сторінку з помилкою реєстрації, де описано причину такої ситуації. Інакше, дані переходять да наступного етапу перевірки – перевірка полів вводу на їх наповненість. Якщо при реєстрації не заповнене поле, або пароль та підтвердження пароля співпадають, то виконується переправлення на сторінку з помилкою наповненості полів. При успішній реєстрації, користувач добавляється в базу даних всіх користувачів, йому видаються права (user). Це означає, що користувач з такими правами може відвідувати сторінки призначені для звичайних користувачів. Тобто їм дається можливість бачити список книг, сортувати їх, фільтрувати, знаходити бажані книги, поповнювати особистий рахунок, завантажувати, конвертувати, читати онлайн. Після завершення бажаних операцій, користувач може покинути власний кабінет і завершити сесію. Для повторного

входу до власного кабінету потрібно пройти наступні етапи: автентифікація і авторизація.

3.1.2 Алгоритм автентифікації

Процедура, яка перевіряє, чи доступна для користувача інформація з пред'явленим ідентифікатором (в нашому випадку логіном і паролем). Це здійснюється за допомогою можливостей веб-серверу Apache Tomcat EE, в якому міститься файл налаштувань (server.xml). А саме, в програмному коді (рис. 3.1).

```
<Realm  className="org.apache.catalina.realm.JDBCRealm"
        driverName="com.mysql.jdbc.Driver"
        connectionURL="jdbc:mysql://localhost/mkr?useUnicode=true&amp;useJDBCCompliantTimezoneShi
        connectionName="root"
        connectionPassword="root"
        userTable="users"
        userNameCol="username"
        userCredCol="password"
        userRoleTable="users_roles"
        roleNameCol="rolename" />

<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

Рисунок 3.1 – server.xml

Де в drivername вказується драйвер для з'єднання з базою даних MySQL, в connectonURL – посилання саме на базу, де записані всі користувачі: в connectionName і connectionPassword логін і пароль до БД; userTable, userNameCol І userCredCol - таблиця з даними про логін і пароль користувачів відповідно; userRoleTable і roleNameCol – таблиця з ролями для користувачів бібліотеки. Система отримує введений користувачем логін і пароль, перевіряє на співпадання з даними в бд, у випадку відсутності – повідомляє, що такого користувача немає, при успішній перевірці - виконується алгоритм авторизації.

3.1.3 Алгоритм авторизації

Процес який дозволяє користувачу отримати доступ до ресурсу у відповідності з його правами доступу, мати відповідний інтерфейс взаємодії з системою, управління своїм власним рахунком, налаштування, редагування та використання представлених можливостей кожному користувачеві.

Для виконання завдання створено 3 види прав доступу: user (звичайний користувач, який може зареєструватися в системі і система відразу присвоїть йому ці права і його функціонал: користуватися наданими ресурсами, завантажувати їх, конвертувати, читати онлайн, поповнювати власний рахунок), content-manager (права які надаються тільки одному користувачеві (можливо також введення інших через запис в БД користувачів), якому представлені можливості управління ресурсом «Бібліотека вільно доступу»: створення нових записів, редагування існуючих, видалення існуючих записів, а також, доступ до сторінки звичайного користувача, для розуміння того, що бачить користувач), admin (адміністратор системи, який має доступ до всіх сторінок і може виконувати операції контент-менеджера, а також свої – видаляти користувачів системи і контент менеджерів)

У разі успішної авторизації, користувачу видаються права, з якими він може спокійно користуватися системою, зважаючи на його роль в цій системі.

3.1.4 Алгоритм поповнення рахунку

При реєстрації рахунок складає 0.0 умовних одиниць. Ця функція створена для умовного відтворення поповнення балансу користувача, яка необхідна для економічної і фінансової забезпеченості розробників та адміністраторів проекту.

Зважаючи на назву ресурсу «Free Access Library», це означає, що доступ до ресурсу через авторизацію має бути безкоштовним, вільним, доступним для

кожного користувача Інтернету, який пройшов реєстрацію і бажає переглянути доступні матеріали. Йому не потрібно вносити оплату, щоб доступитися, відвідати сам ресурс. Він має вільний, необмежений доступ до функцій і час перебування в системі «Бібліотека вільного доступу». Зауважую, що самі книги можуть бути як і безкоштовні (не потребують оплати для завантаження, конвертації чи читання онлайн), так і платні (за які потрібно ввести вказану суму, щоб отримати доступ до представлених функцій і операцій над ними). Це пояснюється тим, що безкоштовними можуть бути - на прикладі фірми: в неї є документація, допоміжні читабельні матеріали, книги, технічна література, професіональна література і все це не потребує оплати зі сторони працівників цієї фірми. Вони маю право вільного доступу до знань, можливості користуватися представленими фірмою матеріалами у вигляді .pdf файлів і застосовувати їх у своїх цілях для покращення продуктивності і знаходження рішення для виконання цілей. Однак, є платні матеріали (патенти, авторські книги які нещодавно вийшли в обіг і інші.), за які сама фірма, або підприємство, або бібліотека заплатили кошти, щоб вони були доступні для користувачів системи. І з економічно-фінансової точки зору вони мають право встановити свою узгоджену ціну, щоб не втрачати власних коштів. Для цього в кабінеті кожного користувача є можливість поповнити власний рахунок. Він здійснюється введенням суми, яка після натискання кнопки зараховується в БД для користувача з відповідним логіном.

3.1.5 Алгоритм отримання балансу користувача

Баланс користувача є важливою функцією, якщо це стосується платних матеріалів бібліотека тому, що гроші важливі для кожної людини і система має коректно відображати і оперувати рахунком. При отриманні поточного балансу,

визивається функція з класу-контролера, які відповідає за відображення коштів. Вона передає ім’я користувача в функцію пошуку користувача за ім’ям і повертає його кількість умовних одиниць.

3.1.6 Алгоритм завантаження матеріалів у форматі .pdf

Для завантаження матеріалів, потрібно ввести індекс книги (документа, іншого виду читального об’єкта) у відповідне поле «id». Після чого, виконується функція перевірки наявності вказаної книги по «id». Якщо такої немає, то користувачу виводиться повідомлення про відсутність такої, на даний момент. Інакше, книга перевіряється на її вартість, якщо вона платна – знімається відповідна сума (якщо в користувача такого суми недостатньо, йому також виводиться повідомлення про це і пропонується поповнити), якщо безплатна – не знімаються кошти. Після цих операцій, шукаємо файл на сервері. Якщо він там відсутній, кошти повертаються назад і виводиться повідомлення про відсутність файлу на сервері. Коли файл знайдено, перевіряється що саме потрібно зробити з ним (конвертувати, чи просто завантажити). При простому завантаженні, в функцію передається файл і атрибут , для того щоб його завантажити.

Функція завантаження (рис. 3.2), дозволяє завантажувати файли розміром до 200 МВ, що реалізовано з точки зору безпеки та надійності веб-додатку (за потреби можна збільшити). За допомогою Response ми повідомляємо запиту, що хочемо завантажити файл у форматі .pdf. Читаємо довжину файлу, записуємо в буфер обміну інформації, очищуємо його після використання, та повідомляємо запит про завершення операції. Після цих дій, у користувача з’являється вікно, що повідомляє його про місце завантаження обраного матеріалу.

```
private void downloadWithValue(File file, String value) throws IOException {

    int readingData;
    final int myAppBuffSize = 209715200;//200 MB
    byte[] creatingBuffer;
    BufferedInputStream bufferedInputStream = null;
    BufferedOutputStream bufferedOutputStream = null;

    response.reset();
    response.setContentType("application/pdf");
    response.setContentLength((int) file.length());
    response.setHeader( s: "Content-disposition",  s1: value+"; filename=" + file.getName());

    try{
        bufferedInputStream = new BufferedInputStream(new FileInputStream(file), myAppBuffSize);
        bufferedOutputStream = new BufferedOutputStream(response.getOutputStream(), myAppBuffSize);
        creatingBuffer = new byte[myAppBuffSize];

        while ((readingData = bufferedInputStream.read(creatingBuffer)) > 0) {
            bufferedOutputStream.write(creatingBuffer,  off: 0, readingData);
        }

        bufferedOutputStream.flush();

    } catch (IOException e) {
        e.printStackTrace();
    }finally {
        Objects.requireNonNull(bufferedInputStream).close();
        Objects.requireNonNull(bufferedOutputStream).close();
    }
    context.responseComplete();
}
```

Рисунок 3.2 – функція завантаження

3.1.7 Алгоритм конвертування з .pdf в .txt

Перед конвертацією виконуються той самий алгоритм перевірки, що і для .pdf, але замість виклику функції завантаження, викликається функція конвертування в .txt, та передається атрибут, що потрібен буде для майбутнього завантаження. Так, як користувачів багато і вже хтось виконував функцію конвертування в .txt, то перевіряємо, чи на сервері вже є такий файл з такою назвою, інакше переходимо до наступних дій. Якщо такого файлу ще немає, то виконуємо конвертацію. Для конвертації була використана існуюча бібліотека класу PDDocument з функцією .load, яка завантажує дані .pdf файлу в змінну типу PDDocument.

Після чого ми використовуємо клас PDFTextStripper і функцію .getText(), в яку передаємо змінну типу PDDocument з отриманими даними. Функція .getText(), перетворює дані в документ .txt і записує в файл (з тією ж назвою тільки іншим форматом). Після чого, викликається функція завантаження (див. рис. 3.2) Алгоритм конвертування з .pdf в .txt (рис. 3.3)

```
private void convertPDFtoTXT(File file) throws IOException {

    File txtFile = new File( pathname: filePath+file.getName().substring(0,file.getName().indexOf("."))
            + Format.TXT.getFormat());

    if (!searchConvertedFile(filePath, txtFile)) {

        PDDocument newPdDocument = PDDocument.load(file);

        if (!newPdDocument.isEncrypted()) {

            PDFTextStripper pdfTextStripper = new PDFTextStripper();
            String text = pdfTextStripper.getText(newPdDocument);
            FileWriter fileWriter = new FileWriter(txtFile);
            fileWriter.write(text);
            fileWriter.close();
        }
        newPdDocument.close();
    }
    downloadWithValue(txtFile, value: "attachment");
}
```

Рисунок 3.3 – алгоритм конвертування .pdf в .txt

3.1.8 Алгоритм конвертування з .pdf в .doc

Перед конвертацією виконуються той самий алгоритм перевірки, що і для .pdf, але замість виклику функції завантаження, викликається функція конвертування в .doc, та передається атрибут, що потрібен буде для майбутнього завантаження. Так, як користувачів багато і вже хтось виконував функцію конвертування в .doc, то перевіряємо, чи на сервері вже є такий файл з такою назвою, інакше переходимо до наступних дій. Якщо такого файлу ще немає, то виконуємо конвертацію.

Для конвертації була використана існуюча бібліотека класу PdfDocument з функцією .loadFromFile(), яка завантажує дані .pdf файлу в змінну типу PdfDocument. Після чого, перетворює в документ формату .doc і, за допомогою функції .saveToFile() записує в файл (з тією ж назвою тільки форматом .doc). Після чого, викликається функція завантаження (див. рис. 3.2) Алгоритм конвертування з .pdf в .doc (рис. 3.4).

```
private void convertPDFtoDOC(File file) throws IOException {

    String link;

    link=filePath+file.getName().substring(0,file.getName().indexOf(".")) + Format.DOC.getFormat();

    File docFile = new File(link);

    if (!searchConvertedFile(filePath, docFile)) {
        PdfDocument docDocument = new PdfDocument();
        docDocument.loadFromFile( filename: filePath+file.getName());
        docDocument.saveToFile(link, FileFormat.DOC);
        docDocument.close();
    }
    downloadWithValue(docFile, value: "attachment");
}
```

Рисунок 3.4 - алгоритм конвертування .pdf в .doc

3.1.9 Алгоритм сортування і фільтрування (пошуку) даних в БД

Користувачам ресурсу доступна функція сортування і пошуку за вказаними властивостями. Як сортувати, так і шукати можна по всім критеріям, причому, шукати можна одночасно за декількома. Результатом такого пошуку будуть дані, які мають схожі дані одночасно за всіма критеріями пошуку. Така можливість відкриває для користувачів максимальну результативність, що полегшує можливість знайти бажане використовуючи тільки ту інформацію, якою вони володіють.

При натисканні, або заповненні полів для сортування (пошуку) відповідно, дані посилаються на функції відображення інформації, яка надходить туди, завдяки

властивостям допоміжного елементу створення інтерфейсу користувача – PrimeFaces. Дані записуються в список. Функції .getSortedBooks() і getFilteredBooks() (рис. 3.5).

```
@ManagedBean
@ViewScoped
public class ManagerController implements Serializable {

    private List<Book> books;
    private List<Book> filteredBooks;
    private List<Book> sortedBooks;
    private BookService bookService = new BookService();

    public List<Book> getSortedBooks() { return books; }
    public List<Book> getFilteredBooks() { return filteredBooks; }
    public void setFilteredBooks(List<Book> filteredBooks) { this.filteredBooks = filteredBooks; }
    public void setSortedBooks(List<Book> sortedBooks) { this.sortedBooks = sortedBooks; }
    public List<Book> getBooks(){
        books=bookService.getAllData();
        return books;
    }
```

Рисунок 3.5 – функції сортування та фільтрування

3.1.10 Алгоритм читання онлайн

Схожий до алгоритму завантаження у форматі .pdf, тільки замість передачі користувачу файлу для завантаження, його перенаправляє за посиланням, де він може читати книгу онлайн і завантажити її за своїм бажанням. Це виконується за допомогою атрибута «inline» (рис. 3.6)

```
<h:commandButton value="Read Book Online"
                 action="#{userController.checkForDownloadWithValue(null,'inline')}"
                 onclick="return confirm('Are you sure? ')"/>
</h:panelGrid>
```

Рисунок 3.6 – атрибут читання онлайн

3.1.11 Алгоритм відображення даних бібліотеки

Щоб відобразити дані бібліотеки після авторизації користувача, нам потрібна функція, яка буде виконуватися самостійно, щоб користувачу не приходилося натискати на кнопку відображення, визиваючи цим відображення даних. В цьому допоможе f:event. Ця конструкція завантажить дані до того як сторінка HTML буде відображена. Тому, що дані ще не доступні для анотації @PostConstruct, яка може виконувати дії після ініціалізації управляючого сторінкою біна (@ManagedBean).

Алгоритм зображено (рис. 3.7).

```
<f:event type="preRenderView" listener="#{managerController.beforeLoadingPage}"/>
```

```
public void beforeLoadingPage(){
    if (!FacesContext.getCurrentInstance().isPostback()){
        getBooks();
    }
}
```

Рисунок 3.7 – алгоритм відображення даних книги

3.1.12 Алгоритм видалення книги

Щоб видалити книгу, контент менеджеру (саме він і адміністратор можуть це зробити), потрібно знайти бажану книгу по параметрам та натиснути кнопку видалення. Після чого виводиться повідомлення, чи дійсно він планує виконати цю дію, і вже потім викликається метод .deleteBook(). Який визиває метод сервісу видалення книги (bookServise.deleteData()) за вказаним «id» та «name». Сервіс видаляє всі формати книги з вказаною назвою в папці (uploads), де зберігаються файли і викликає функцію видалення книги з бази даних. Яка в свою чергу, за

допомогою технології запитів до бази даних (JPA Hibernate), видаляє книгу (документ) і її інформацію з БД.

Детально зображено (рис. 3.8).

```
public void deleteBook(int id,String name) throws IOException {
    FacesContext context = FacesContext.getCurrentInstance();
    bookService.deleteData(id,name);
    context.getExternalContext().redirect( s: "http://localhost:8080/manager_pages/manager_hello.xhtml");
}
```

```
public void deleteData(int id,String name) {

    String pathname="d:/uploads/";
    bookDAO.deleteData(id);

    for(Format format : Format.values()) {
        if (new File( pathname: pathname+name + format.getFormat()).delete()) {
            System.out.println("Все файлы с именем "+name+" удалены");
        }
    }
}
```

```
public void deleteData(int id) {

    try {
        BookEntityMF bookEntityMF = new BookEntityMF();
        EntityManager entityManager = bookEntityMF.getEntityManager();
        Book findBook = entityManager.find(Book.class, id);
        //удаление
        entityManager.getTransaction().begin();
        entityManager.remove(findBook);
        entityManager.getTransaction().commit();
        entityManager.close();
        bookEntityMF.close();
        System.out.println("Объект удалён.");
    }catch (IllegalArgumentException e){
        throw new RuntimeException("Такого id не существует.");
    }
}
```

Рисунок 3.8 – алгоритм видалення книги

3.1.13 Алгоритм додавання книги в бібліотеку

Щоб додати книгу в бібліотеку, потрібно заповнити дані в формі вводу, для яких накладені обмеження (вони мають бути заповнені, форми для чисел мають відповідати числам, завантажити файл і картинку) після чого натиснути відповідну кнопку. Запит визиває AddBookServlet (сервлет додавання книги), в якому зчитуються дані з полів вводу, переводяться у відповідний формат та присвоюються змінним. Створюється папка для майбутніх файлів де вони будуть зберігатися (при додаванні першої книги), якщо така вже є то файли продовжують туди записуватися.

За допомогою функції extractFileName() (рис. 3.9)

```
//Извлекаем имя файлов
private String extractFileName(Part part) {

    String fileName = "";
    String contentDisposition = part.getHeader( s: "content-disposition");
    String[] items = contentDisposition.split( regex: ";");
    for (String item : items) {
        if (item.trim().startsWith("filename")) {
            fileName = item.substring(item.indexOf("=") + 2, item.length() - 1);
            newBookName=fileName;
        }
    }
    return fileName;
}
```

Рисунок 3.9 – функція вилучення ім’я

вилучаємо ім’я файлу, що завантажився, обробляючи, прийнятий сервлетом файл класу Part. Тепер проводимо операцію з вилучення картинки, яку добавив менеджер, і перетворення її у String, за допомогою шифрування Base64. Після отримання назви файлу і закодовоної картинки переходимо до додавання книги в існуючу бібліотеку. Алгоритм додавання зображено на (рис. 3.10).

```
public void addData(Book book) {

    try {
        BookEntityMF bookEntityMF =new BookEntityMF();
        EntityManager entityManager = bookEntityMF.getEntityManager();
        entityManager.getTransaction().begin();
        entityManager.persist(book);
        entityManager.getTransaction().commit();
        entityManager.close();
        bookEntityMF.close();
        System.out.println("Объект сохранён.");
    }catch (PersistenceException e){
        throw new RuntimeException("Проверте корректность вводимых данн
    }
}
```

```
public void addData(Book book) {
    bookDAO.addData(book);
}
```

Рисунок 3.10 – алгоритм додавання

3.1.14 Алгоритм зміни розміру картинки

Щоб відобразити картинку в таблиці на сайті, потрібно вказати її розмір. Так як, контент менеджер може додавати абсолютно різні по розміру картинки .jpg, вони не будуть відповідати одна одній, що призведе до неправильно відображення їх на сайті. Для цього ми створили функцію resizeImage(), яка буде усі вхідні картинки підводити до стандартного вказаного розміру. Детальний алгоритм зображено на (рис. 3.11)

```
private BufferedImage resizeImage(BufferedImage bufferedImage, int newWidth, int newHeight) {

    int resizedWidth = bufferedImage.getWidth();
    int resizedHeight = bufferedImage.getHeight();
    BufferedImage resizedImage = new BufferedImage(newWidth, newHeight, bufferedImage.getType());
    Graphics2D graphics2D = resizedImage.createGraphics();
    graphics2D.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
    graphics2D.drawImage(bufferedImage,  dx1: 0,  dy1: 0, newWidth, newHeight,  sx1: 0,  sy1: 0, resizedWidth, resizedHeigh
    graphics2D.dispose();
    return resizedImage;
}
```

Рисунок 3.11 – алгоритм зміни розміру картинки

3.1.15 Алгоритм шифрування картинки для запису в БД

Щоб зберегти картинку в базі даних, обрано символьний формат типу String. Функція encodeImage() (рис. 3.12) приймає на вхід тип Part (записана в буфер картинка, яку відправив контент менеджер), за допомогою класу InputStream записує в змінну inputStream картинку побайтово. Після чого, за допомогою класу BufferedImage, записує інформацію у змінну byfferedImage. За допомогою цього класу, ми зашифровуємо змінену у розмірі картинку в текстовий формат технологією Base64.

```
String encodeImage(Part partImage) throws IOException {

    //Кодируем image и переопределяем в удобный нам размер
    InputStream inputStream =partImage.getInputStream();
    BufferedImage bufferedImage = ImageIO.read(inputStream);
    BufferedImage resizedImg = resizeImage(bufferedImage, newWidth: 180, newHeight: 280);
    ByteArrayOutputStream imageByteArrayOutputStream = new ByteArrayOutputStream();
    ImageIO.write(resizedImg, formatName: "jpg", imageByteArrayOutputStream);
    return Base64.getEncoder().encodeToString(imageByteArrayOutputStream.toByteArray());
}
```

Рисунок 3.12 – алгоритм шифрування картинки

3.1.16 Алгоритм редагування картинки

Після того, як контент-менеджер натисне на кнопку редагування, запит відправиться до сервлету EditBookServlet методом POST. В сервлеті дані зчитуються з полів вводу, зчитується картинка (для оновлення аватарки). Далі перевіряється чи присутня така книга в бібліотеці за вказаним «id». Якщо не знайдено, виводиться повідомлення про відсутність такої книги за казаним «id». Інакше, шифрується за допомогою Base64 (див. рис. 3.12). Викликається функція сервісу оновлення

даних в БД (рис. 3.13). В якій змінюється назва книги в папці для всіх форматів, а також змінюються інші дані в базі даних MySQL. Після чого, оновлюється сторінка з відкоригованою інформацією.

```
private void editBook(Book editBook, int id){

    Book book;
    String oldLink;
    BookService bookService=new BookService();

    book=bookService.findBookById(id);
    oldLink=book.getLink();
```

Рисунок 3.13 – редагування картинки

```
bookService.updateData(editBook,id);

for(Format format : Format.values()) {
    if (new File( pathname: oldLink + format.getFormat()).renameTo(new File( pathname: editBook.getLink() + format.getFormat()))) {
        System.out.println("Файл "+oldLink+ format.getFormat()+" переименован в "+editBook.getLink() + format.getFormat());
    }else {
        System.out.println("Переименовать не получилось...");
    }
```

Рисунок 3.13 – редагування картинки (продовження)

3.1.17 Детальніше про алгоритм Base64

Base64 – метод кодування при якому набір байтів нашої картинки перетворюються в строку з ASCII символів (ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg hijklmnopqrstuvwxyz0123456789+/=). Таких символів 64. Алгоритм бере кожних 3 байта (24 біта) картинки і переводить в один з цих символів. Далі виділяються групи по 6 бітів і переводяться в десятинну систему обчислення, де з групи по 8 бітів вибирається символ з ASCII. При отриманні картинки в браузері, нам не обов’язково її розшифровувати, за нас це робить сам браузер тому, що він

підтримує стандарт RFC 2397. Сенс стандарту в тому, що формат виводу (в нашому випадку - картинки) має такий вид:

data:[type][;base64],data

(код зашифрованої картинки)

3.1.18 Алгоритм видалення користувачів адміністратором

Функція deleteUser() доступна тільки адміністратору і вона видаляє користувача бібліотеки з бази даних за вказаним логіном (ім'ям), а також, видаляє його права доступу, що дає можливість контролювати користувачів системи.

3.2 Опис системи

Система реалізовувалась з використанням технології JSF [7]. Це платформа веб-додатків, реалізована на Java, запропонована для полегшення розробки інтерфейсу користувачів для веб-ресурсів. Компоненти, які служать для вводу даних, можуть повторно використовуватись, що зменшує кількість коду для розробників. Технологія дозволяє поєднати дані проекту, інтерфейс користувача та процес обробки запитів на серверній частині. Представляє інтерфейс для розробки компонентів та структури сторінок. Є можливість використовувати стандартні компоненти, доповнювати їх, а також створювати нові, які інтегруються без особливих зусиль. Їх легко використовувати, очевидно та практично. Головним плюсом є те, що через компоненти передаються запити на сервер і вже на серверній частині виконуються основні операції і вся бізнес-логіка, що покращує читабельність коду, програми і робить її простою в розумінні. Не так, як з JSP, де код Java і HTML перемішуються в одне ціле, і при масштабному, складному проекті буде

нереально зрозуміти як краще налаштувати, оптимізувати, покращити програму, а тим більше, передати проект іншим розробникам. Присутня велика бібліотека тегів, які включають в собі всі необхідні функції, за допомогою яких, можна робити справді якісні, продуктивні та сучасні веб-сервіси. Вбудовані функції, що також зменшують кількість коду і затрату часу на розробку компонентів, які вже готові і потребують лиш підключення (сортування, фільтр, редагування, таблиці, вивід динамічних таблиць, текстові поля з динамічним відображенням). Легко підлаштовується під управління сервлетами та бінами.

Сервлети є важливою частиною нашого проекту. Унаслідуються від класу HttpServlets, розширяють функції цього класу та запускаються всередині контейнеру сервлетів (наприклад Apache Tomcat EE). Обробляють інформацію рядом методів за вказаною адресою, яка пишеться в анотації сервлету (зазвичай за назвою самого сервлету). GET запити в нашій програмі, повідомляють користувача про помилки, з якими він може зіткнутися під час використання веб-додатку, вони призначення для отримання даних. POST запити навпаки – для відправлення даних користувачем. Для прикладу, коли менеджер хоче додати нову книгу, або редагувати присутню, чи видалити її – він викликає запит POST і передає дані серверній частині для подальших дій. HttpServletRequest (один з параметрів функцій сервлета) зберігає інформацію про запит, HttpServletResponse відправляє відповідь на запит. Потрібно зауважити, що при відправці сервлету запитів, створюється лиш одна його копія. А при тривалій бездіяльності, контейнер сервлетів завершає роботу невикористовуваних сервлетів. Дані в форму передаються і отримуються за допомогою методів getParametr(). Також, за допомогою sendRedirect() можна перенаправляти запит на іншу сторінку, або оновлювати її. Що допоможе нам для реалізації коректних функцій та виведенні потрібної інформації. Важливою властивістю сервлета в нашому веб-додатку є те, що при виході з

кабінету він виконує функцію завершення сесії користувача (вихід з власного кабінету).

ManagedBeans – це звичайні класи Java, яким присвоєна анотація @ManagedBean, дозволяє їх поєднувати з технологією JSF. В класах такого типу реалізовано змінні, що оперують при виводі інформації, даних про книги та функції, які виконують бізнес-логіку проекту. Їх можна використовувати в файлах XHTML. Завдяки такому підходу класами легко керувати, можна зручно доповнювати (дописавши потрібну функцію і добавивши для неї інтерфейс) та отримувати з

них інформацію (дані). Такі класи знадобилися нам при створенні керуючих класів для інтерфейсів користувача, контент-менеджера, адміністратора.

Висновок до розділу 3

Створено 3 ролі. При реєстрації кожному користувачу видаються права користувача, які дозволяють йому відвідувати ресурс, бачити базу даних книг, читати їх онлайн, завантажувати, конвертувати, поповнювати свій баланс рахунку. Контент-менеджеру надано можливість керувати бібліотекою, щоб він міг створювати нові книги, завантажувати їх в базу даних, редагувати ті що є в наявності, видаляти непотрібні. Адміністратору видано максимальну кількість прав доступу. Він може керувати книгами (з відсутності контент-менеджера), а також, користувачами, що зареєстровані в веб-додатку «Бібліотека вільного доступу».

Реалізовано алгоритми для виконання завдання веб-додатку, і операцій, які прив’язані до інтерфейсів користувачів.

РОЗДІЛ 4
ІНСТРУКЦІЯ РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

4.1 Звичайний користувач

При вході до ресурсу, вас вітає початкова сторінка, з якої можна потрапити до вікна реєстрації, або входу в систему (рис. 4.1).

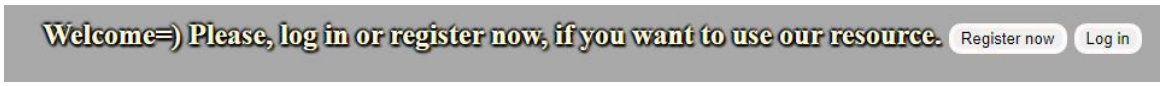


Рисунок 4.1 – вхід в систему

Якщо Ви новий користувач, то для початку роботи з системою потрібно ***зареєструватися***, натиснувши кнопку (рис. 4.2).

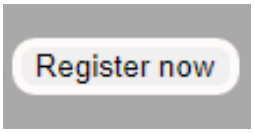


Рисунок 4.2 – реєстрація

Після чого, вас переадресовує на сторінку реєстрації (рис. 4.3), де всі поля повинні бути ***заповнені***: поле для логіну, пароля, повторного пароля. Логін не повинен бути схожим на вже зареєстровані в системі. Інакше, буде повідомлення про помилку.

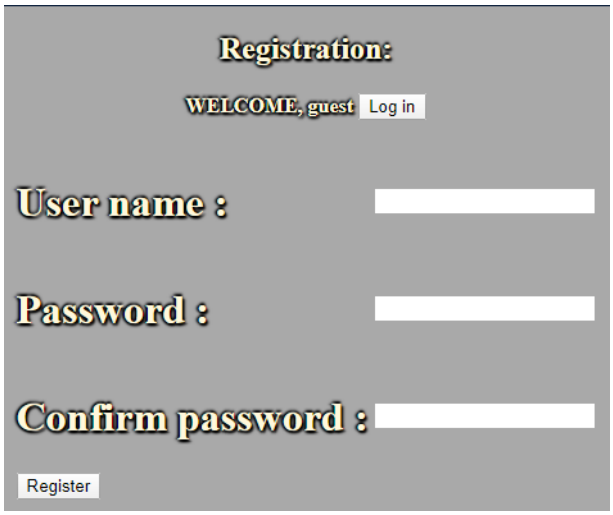


Рисунок 4.3 – сторінка реєстрації

Після заповнення полів потрібно натиснути кнопку «Register». Після чого відбудеться перевірка на коректність і користувач з новим логіном буде зареєстрований в системі і зможе увійти до власного кабінету, натиснувши кнопку «Log in». Відбудеться переадресування на сторінку входу в систему (рис. 4.4).

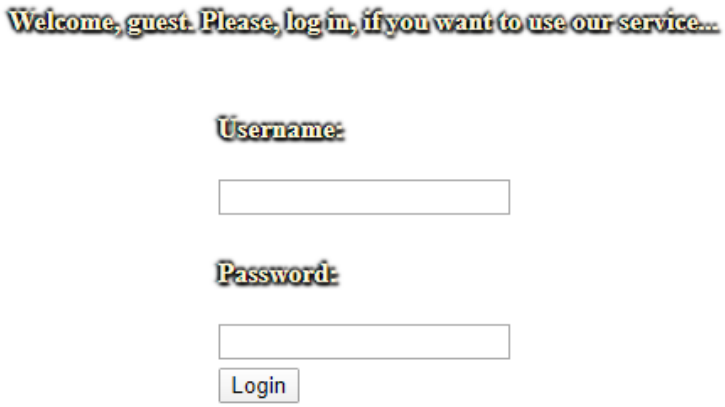


Рисунок 4.4 – вхід в систему

На якій потрібно ввести дані, що вводили при реєстрації. Після вводу логіна і пароля відкривається головна сторінка користувача (рис. 4.5).

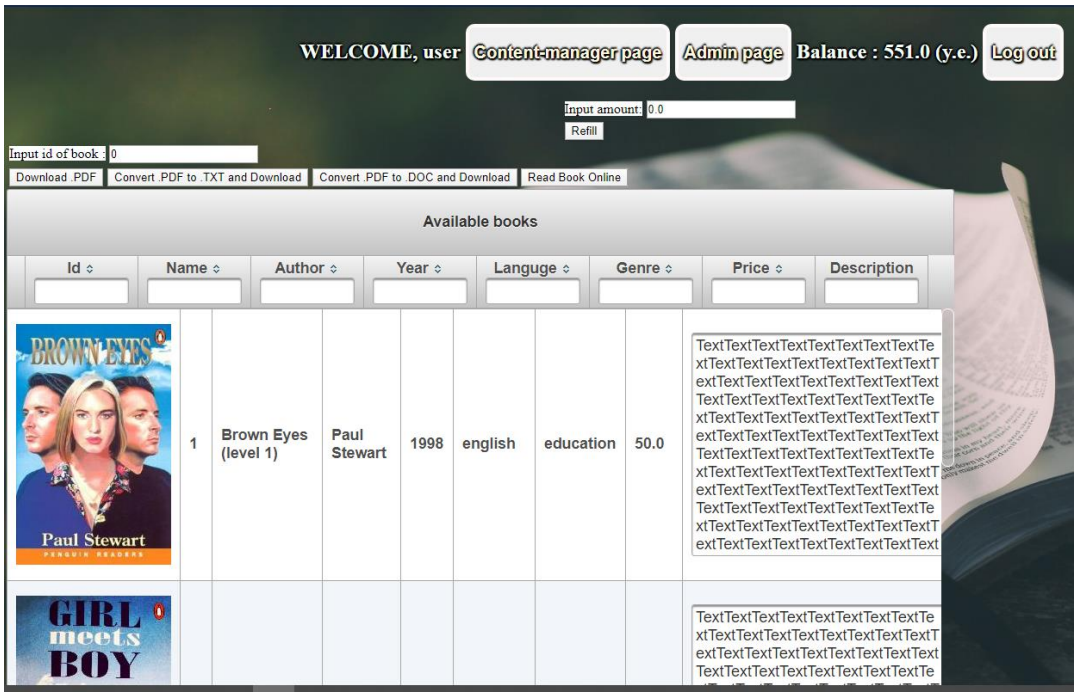


Рисунок 4.5 – головна сторінка користувача

На головній сторінці користувача, можна сортувати книги (рис. 4.6), фільтрувати їх(рис. 4.6), шукати за декількома критеріями (рис. 4.6), читати опис книг та завантажувати в .pdf (натиснувши кнопку «Download .PDF»), конвертувати в .txt (натиснувши кнопку «Convert .PDF to .TXT and Download»), конвертувати в .doc (натиснувши кнопку «Convert .PDF to .DOC and Download»), читати онлайн (натиснувши кнопку «Read Book Online») (див. рис. 4.5), поповнити рахунок (натиснувши кнопку «Refill») і ввівши відповідну суму поповнення.

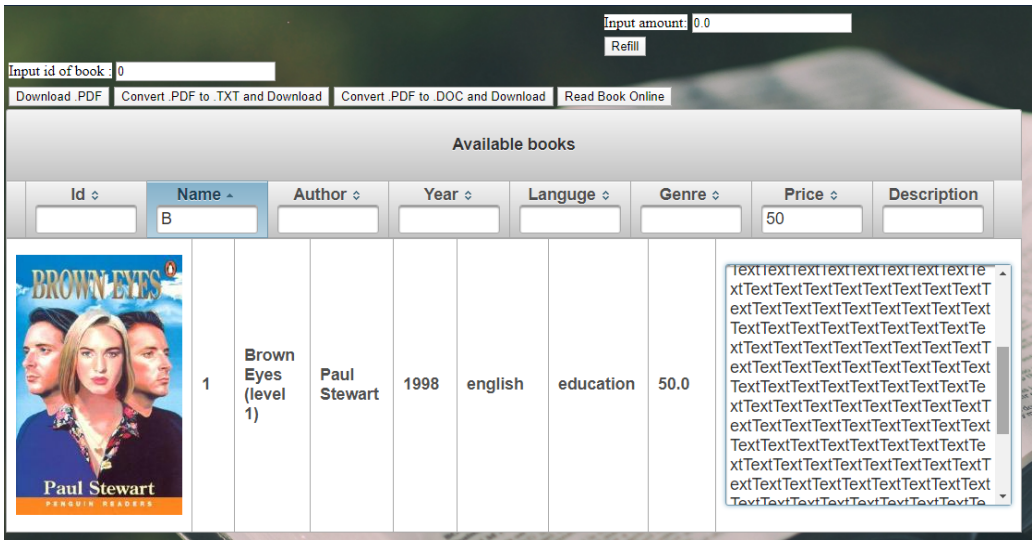


Рисунок 4.6 – функції користувача

Завантажувати, конвертувати, читати онлайн можна після введення відповідного «id».

Щоб покинути кабінет потрібно натиснути кнопку «Log out» (див. рис. 4.5).

Після цього відбудеться розлогування і переадресація на сторінку введення логіна і пароля.

4.2 Контент-менеджер

Дані для входу контент-менеджера зберігаються в базі даних користувачів. Щоб увійти в ролі контент-менеджера потрібно ввести ці дані. Відбудеться переадресування на головну сторінку користувача. Для переходу на сторінку контент-менеджера – натисніть кнопку «Content manager page» (див. рис. 4.5).

Окрім можливостей звичайного користувача для вас відкриється можливість видаляти книгу (рис. 4.6), добавляти нову (рис. 4.7), редагувати (рис. 4.8).

Рисунок 4.6 – видалення книги

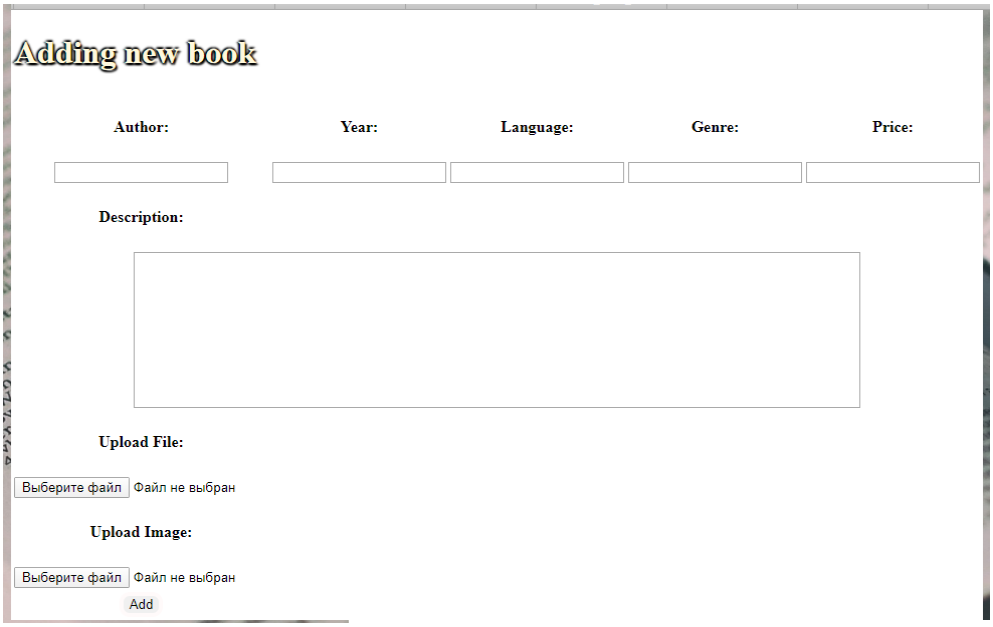


Рисунок 4.7 – додавання книги

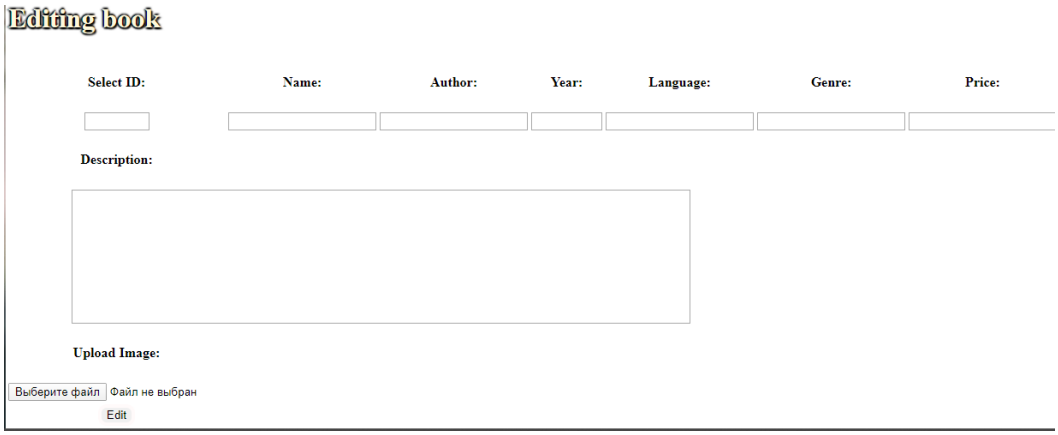


Рисунок 4.8 – редагування книги

Завантажити книгу можна в форматі .pdf і картинку для її аватарки формату .jpg. Редагувати можна тільки ті книги, які присутні в бібліотеці.

4.3 Адміністратор

Адміністратору, крім всіх можливостей, добавлена можливість видаляти користувачів системи (рис. 4.9).

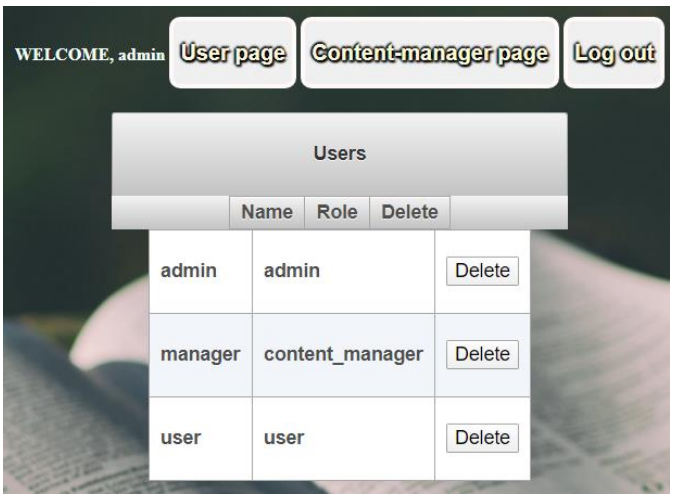


Рисунок 4.9 – видалення користувачів

4.4 Приклад доповнення нової книги

Зайдемо до ресурсу в ролі контент менеджера і добавимо книгу Миколи Гоголя «Шинель» до нашої бібліотеки. Ввівши дані про книгу (рис. 4.10), та переконавшись, що вони збереглися на сервері (рис. 4.11).

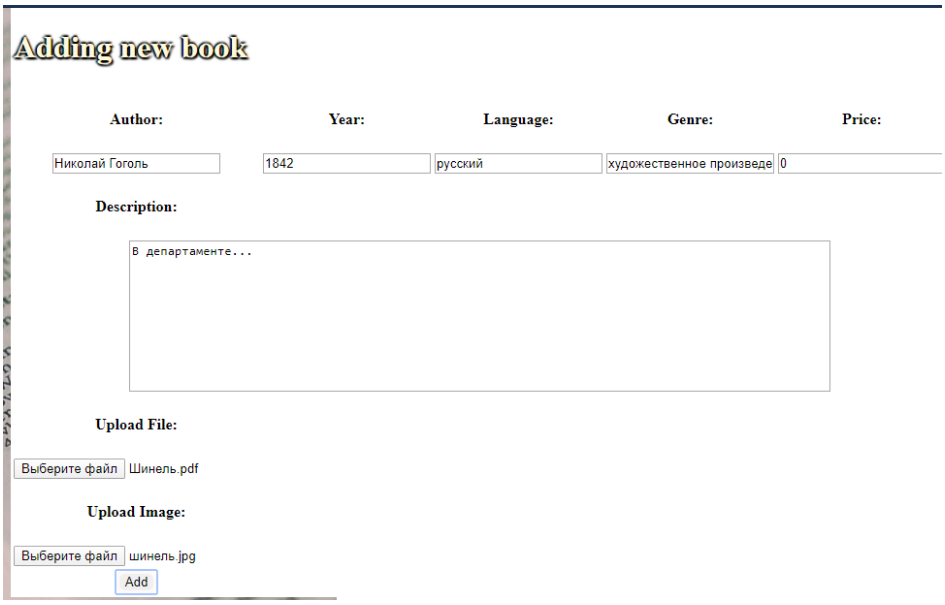


Рисунок 4.10 – введення даних про книгу

Рисунок 4.11 – знайдена книга

4.5 Приклад завантаження

Для завантаження книги, було вказано «id=5», яке відповідає книзі з назвою «Shinel». Результат – виведене вікно вибору директиви для завантаження книги (рис. 4.12), а також, конвертування в інші формати.

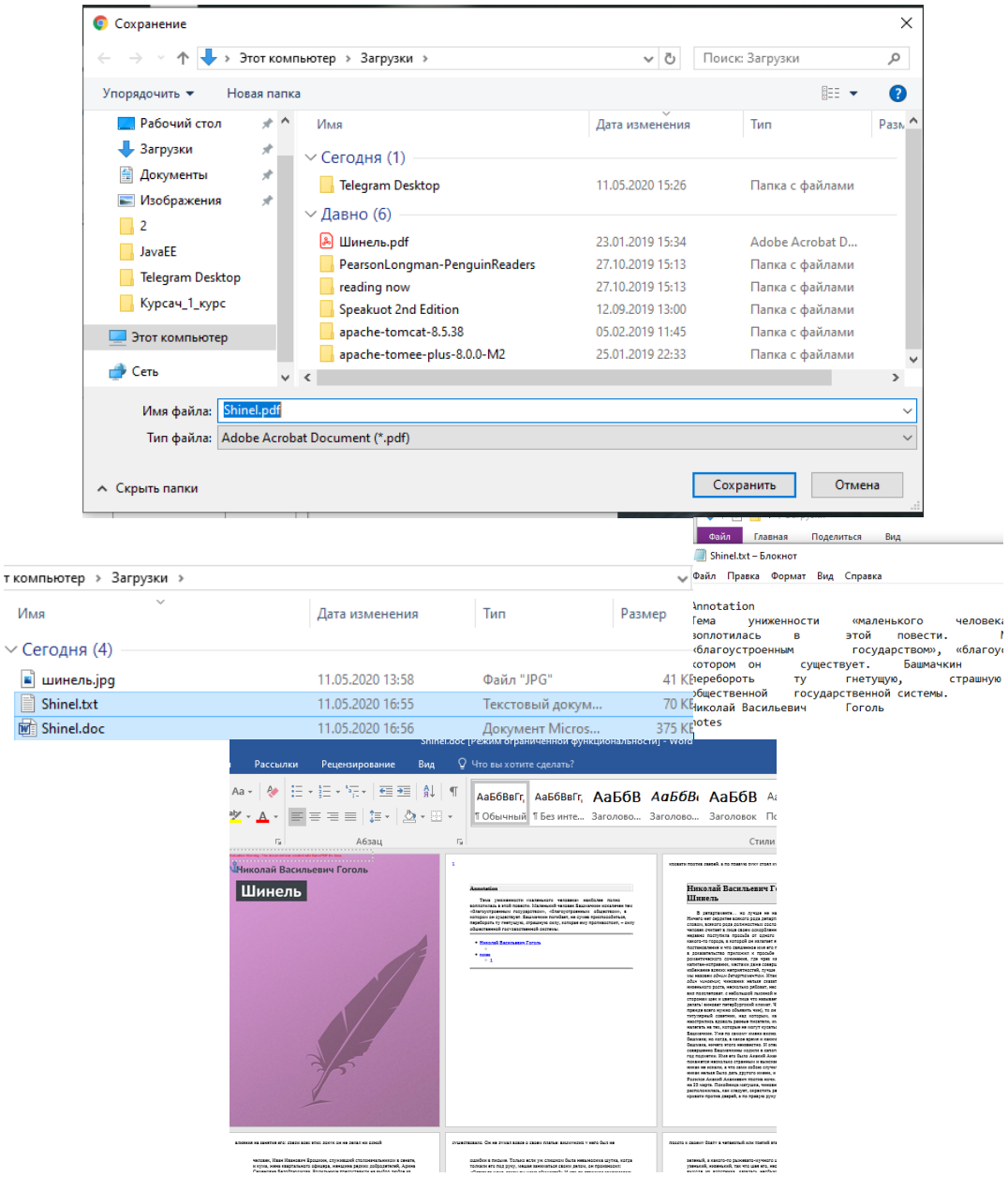


Рисунок 4.12 – приклад завантаження

4.6 Приклад редагування і видалення книг

Для редагування доступних книжок, контент-менеджеру потрібно скористатися таблицею для редагування. Вона заповнюється таким самим методом, як і таблиця додавання інформації. Після чого, слід натиснути кнопку «Edit» (див. рис. 4.8), після чого, дані в таблиці зміняться на нові, а також, назва книги і всі її формати у папці зберігання (uploads).

Для видалення – натиснути кнопку «Delete» (див. рис. 4.6), і дані видаляться з таблиці та всі формати книги з папки зберігання (uploads).

Висновок до розділу 4

Виконано завдання по створенню веб-додатка з функціями, що мають задовольнити потреби користувачів. Всі операції перевірені, протестовані і працюють без помилок. Виправлені помилки, з якими може зіткнутися користувач. Наведено інструкцію з використання додатку «Бібліотека вільного доступу» для всіх можливих ролей та наведено декілька прикладів роботи програми.

ВИСНОВКИ

Отримавши завдання створити веб-ресурс «Бібліотека вільного доступу» (Free Access Library) було проаналізовано сучасний стан актуальності бібліотек, зібрань, наукової літератури, та зроблено висновок, що існуючі ресурси не повністю задовольняють потреби навчальних закладів, підприємств та існуючих бібліотек.

Поставлено завдання розробити більш досконалий, адаптивний для різних галузей веб-додаток. За допомогою якого, користувач зможе бачити список матеріалів для читання, мати можливість їх шукати по різним параметрам, завантажувати їх, конвертувати в потрібний формат, а також, поповнювати баланс, щоб придбати платні видання. Дати можливість адміністраторам ресурсу розвиватися в фінансово-економічній сфері і підтримувати свою бібліотеку.

Перед розробкою веб-додатку, вирішено використовувати сучасні технології, які буде просто і без лишніх затрат змінити на сучасніші. Обрано мову програмування Java для серверної бізнес-логіки; технологію JSF, PrimeFaces, HTML – для розробки інтерфейсу користувача, базу даних MySQL.

Під час розробки було розроблено велику кількість алгоритмів, що потрібні для повної, логічної та безпомилкової реалізації завдання бакалаврського проекту.

Кожен алгоритм пройшов тестування і реалізований належним чином.

Для управління інтерфейсом використано Java-класи з анотацією @ManagedBean, сервлети – для обробки помилок і завантаження даних на сервер контент- менеджером. Також технологію JPA для створення функцій управління базою даних (видалення, оновлення, доповнення, редагування).

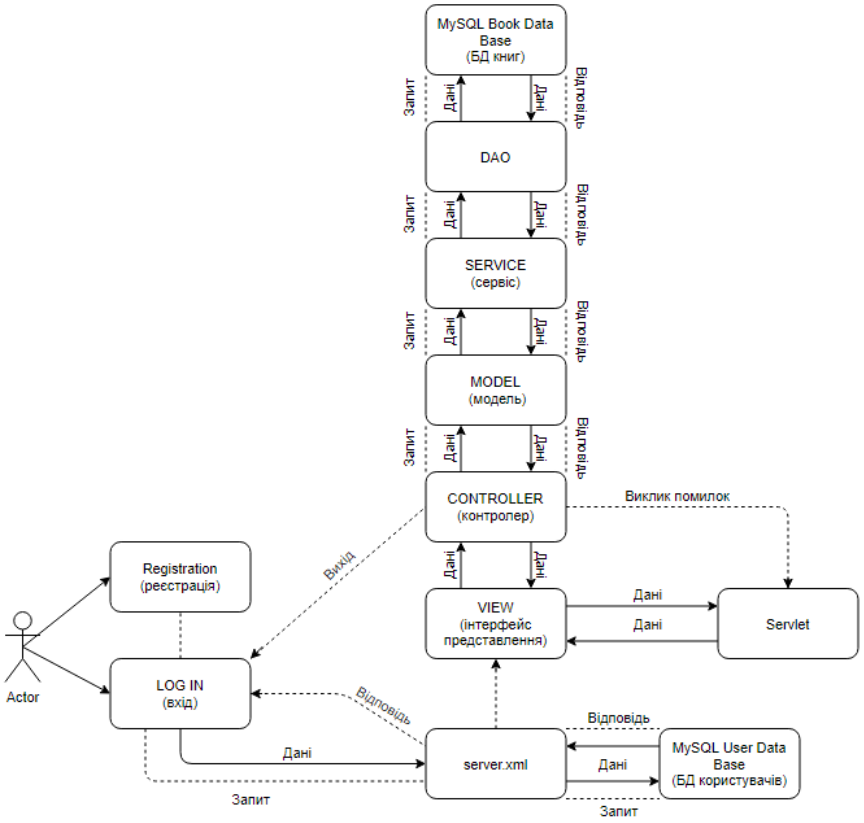
Шаблон проектування MVC для незалежних частин коду і легкості в розумінні, а також, простій взаємодії компонентів і можливості доповнювати програму просто додавши нову функцію з бізнес-логікою і реалізувавши для неї інтерфейс.

Використовуючи програмне забезпечення IntelliJ IDEA 2019.2.3, налагоджено роботу веб-додатку, виправлено помилки, налагоджено роботу і функціонування всіх компонентів в цілому.

Створено інструкцію користувача, в якій можна побачити, що всі завдання бакалаврської розробки були виконані і реалізовані. Програмне забезпечення «Бібліотека вільного доступу» (Free Access Library) готова до використання в цілях створення своєї бібліотеки для університету, школи, підприємства та інших закладів, що потребують організованості та формалізації даних у вигляді бази даних з власним кабінетом користувача.

ЛІТЕРАТУРА

1. Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 с.
2. Эккель, Брюс Философия Java / Брюс Эккель. - М.: Питер, 2016. - 809 с.
3. Гупта, Арун Java EE 7. Основы / Арун Гупта. - М.: Вильямс, 2014. - 336 с.
4. Гонсалвес, Энтони Изучаем Java EE 7 / Энтони Гонсалвес. - М.: Питер, 2016. - 640 с.
5. Хорстманн, Кей С. Java SE 8. Вводный курс / Хорстманн Кей С.. - М.: Диалектика / Вильямс, 2014. - 898 с.
6. [Stackoverflow]. URL: https://stackoverflow.com/
7. [Metanit]. URL: https://metanit.com/java/javaee/
8. [Tomee]. URL: http://tomee.apache.org/apache-tomee.html
9. [Primefaces]. URL: https://www.primefaces.org/



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Лейзьо С. І.		
Перевір.		Сімоненко В.П		
Н. контр.		Сімоненко В.П.		
Затверд.		Стіренко С.Г.		

ДП 6412. 03.000 Д1

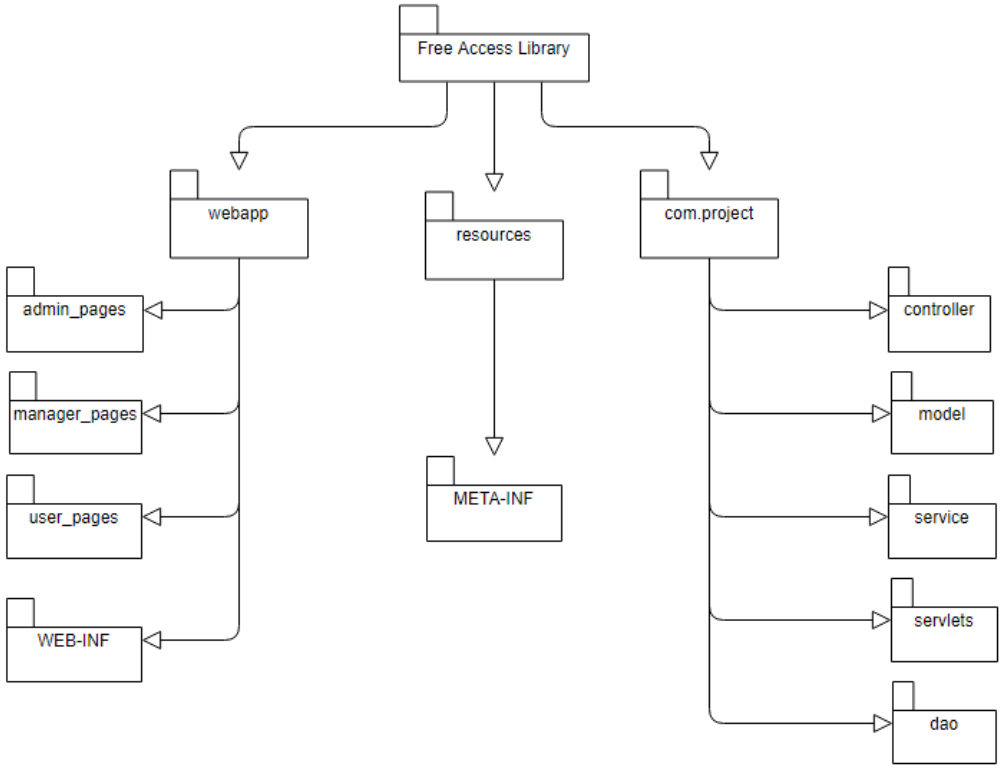
Бібліотека вільного доступу
Функціональна схема

Літ.	Маса	Масштаб

Арк. 1 Аркушів 1

Дипломний проект

КПІ ФІОТ
кафедра ОТ
гр. ІП-64



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Лейзьо С. І.		
Перевір.		Сімоненко В.П		
Н. контр.		Сімоненко В.П.		
Затверд.		Стіренко С.Г.		

ДП 6412. 04.000 Д2

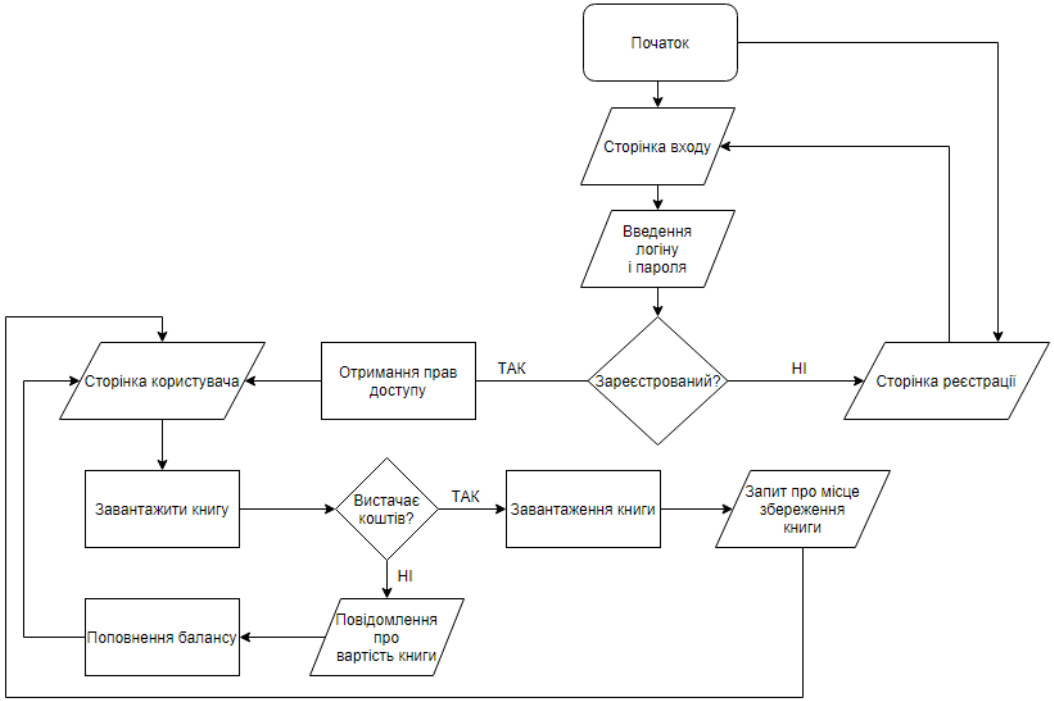
Бібліотека вільного доступу
Структурна схема

Літ.	Маса	Масштаб

Арк. 1 Аркушів 1

Дипломний проект

КПІ ФІОТ
кафедра ОТ
гр. ІП-64



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Лейзьо С. І.		
Перевір.		Сімоненко В.П		
Н. контр.		Сімоненко В.П.		
Затверд.		Стіренко С.Г.		

ДП 6412. 05.000 Д3

Бібліотека вільного доступу
Алгоритм

Літ.	Маса	Масштаб

Арк. 1 Аркушів 1

Дипломний проект

КПІ ФІОТ
кафедра ОТ
гр. ІП-64

ДОДАТОК А

«Бібліотека вільного доступу»

Текст програми

ДП 6412. 06.000 Б1

Листів 20

Київ – 2020 року

BookDAO

```
package com.lab1.database.dao;

import com.lab1.database.model.Book;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceException;
import javax.persistence.Query;
import java.util.List;

public class BookDAO {

    public List<Book> getAllBooks() {

        BookEntityMF bookEntityMF = new BookEntityMF();
        EntityManager entityManager = bookEntityMF.getEntityManager();
        Query query = entityManager.createQuery("SELECT e FROM Book e");
        List<Book> book = query.getResultList();
        System.out.println("Данные записались в bookList.");
        entityManager.close();
        bookEntityMF.close();
        return book;
    }

    public void findData(int id) {

        try {
            BookEntityMF bookEntityMF = new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();
            Book book = entityManager.find(Book.class, id);
            System.out.println(book.toString());
            entityManager.close();
            bookEntityMF.close();
            System.out.println("Выполнено.");
        }catch (NullPointerException e){
            throw new RuntimeException("Такого id не существует.");
        }
    }

    public void addData(Book book) {

        try {
            BookEntityMF bookEntityMF =new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();
            entityManager.getTransaction().begin();
            entityManager.persist(book);
            entityManager.getTransaction().commit();
            entityManager.close();
            bookEntityMF.close();
            System.out.println("Объект сохранён.");
        }catch (PersistenceException e){
            throw new RuntimeException("Проверте корректность вводимых данных
```

```
объекта.");
        }
    }

    public void updateData(Book book, int id) {

        try {
            BookEntityMF bookEntityMF = new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();

            Book findBook = entityManager.find(Book.class, id);
            System.out.println(findBook.toString());

            entityManager.getTransaction().begin();
            findBook.setName(book.getName());
            findBook.setAuthor(book.getAuthor());
            findBook.setYear(book.getYear());
            findBook.setLanguage(book.getLanguage());
            findBook.setGenre(book.getGenre());
            findBook.setPrice(book.getPrice());
            findBook.setDescription(book.getDescription());
            findBook.setLink(book.getLink());
            findBook.setEncryptedImage(book.getEncryptedImage());
            entityManager.getTransaction().commit();

            findBook = entityManager.find(Book.class, id);
            entityManager.close();
            bookEntityMF.close();
            System.out.println(findBook.toString());
            System.out.println("Объект обновлён.");
        } catch (NullPointerException e){
            throw new RuntimeException("Такого id не существует.");
        }
    }

    public void deleteData(int id) {

        try {
            BookEntityMF bookEntityMF = new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();
            Book findBook = entityManager.find(Book.class, id);
            //удаление
            entityManager.getTransaction().begin();
            entityManager.remove(findBook);
            entityManager.getTransaction().commit();
            entityManager.close();
            bookEntityMF.close();
            System.out.println("Объект удалён.");
        }catch (IllegalArgumentException e){
            throw new RuntimeException("Такого id не существует.");
        }
    }

    public String findDataById(int id) {
```

```
        String bookName;
        try {
            BookEntityMF bookEntityMF = new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();
            Book book = entityManager.find(Book.class, id);
            bookName=book.getName();
            entityManager.close();
            bookEntityMF.close();
            System.out.println("Выполнено.");
        }catch (NullPointerException e){
            throw new RuntimeException("Такого id не существует.");
        }
        return bookName;
    }

    public Book findBookById(int id) {

        Book findBook;
        try {
            BookEntityMF bookEntityMF = new BookEntityMF();
            EntityManager entityManager = bookEntityMF.getEntityManager();
            findBook = entityManager.find(Book.class, id);
            entityManager.close();
            bookEntityMF.close();
            System.out.println("Выполнено.");
        }catch (NullPointerException e){
            throw new RuntimeException("Такой книги не существует.");
        }
        return findBook;
    }
}
```

BookEntityMF

```
package com.lab1.database.dao;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

class BookEntityMF {

    private EntityManagerFactory emFactory;

    EntityManager getEntityManager(){
        emFactory = Persistence.createEntityManagerFactory("BookPersistenceUnit");
        return emFactory.createEntityManager();
    }

    void close(){
        emFactory.close();
```

```
    }
}
```

UserDAO

```
package com.lab1.database.dao;

import com.lab1.database.model.User;
import com.lab1.database.model.UserRole;

import javax.persistence.EntityManager;
import javax.persistence.Query;
import java.util.List;

public class UserDAO {

    public List<UserRole> getAllUserRole() {

        UsersEntityMF usersEntityMF = new UsersEntityMF();
        EntityManager entityManager = usersEntityMF.getEntityManager();
        Query query = entityManager.createQuery("SELECT e FROM UserRole e");
        List<UserRole> usersList=query.getResultList();
        System.out.println("Данные записались в usersList.");
        entityManager.close();
        usersEntityMF.close();
        return usersList;
    }

    public void add(User newUser, String role) {

        UserRole userRole=new UserRole(newUser.getUsername(),role);
        UsersEntityMF usersEntityMF = new UsersEntityMF();
        EntityManager entityManager = usersEntityMF.getEntityManager();
        entityManager.getTransaction().begin();
        entityManager.persist(newUser);
        entityManager.persist(userRole);
        entityManager.getTransaction().commit();
        entityManager.close();
        usersEntityMF.close();
        System.out.println("Пользователь добавлен в базу.");
    }

    public void updateData(User user, String name) {

        try {
            UsersEntityMF usersEntityMF = new UsersEntityMF();
            EntityManager entityManager = usersEntityMF.getEntityManager();
            User findUser = entityManager.find(User.class, name);
            entityManager.getTransaction().begin();
            findUser.setY_e(user.getY_e());
            entityManager.getTransaction().commit();
            entityManager.close();
```

```
            usersEntityMF.close();
            System.out.println("Объект обновлён.");
        } catch (NullPointerException e){
            throw new RuntimeException("Такого пользователя не существует.");
        }
    }

    public void delete(String username) {

        try {
            UsersEntityMF usersEntityMF = new UsersEntityMF();
            EntityManager entityManager = usersEntityMF.getEntityManager();
            User findUser = entityManager.find(User.class, username);
            UserRole findUserRole = entityManager.find(UserRole.class, username);
            //удаление
            entityManager.getTransaction().begin();
            entityManager.remove(findUserRole);
            entityManager.remove(findUser);
            entityManager.getTransaction().commit();
            entityManager.close();
            usersEntityMF.close();
            System.out.println("Пользователь удалён.");
        }catch (IllegalArgumentException e){
            throw new RuntimeException("Такого пользователя не существует.");
        }
    }

    public User findUserByName(String name) {

        User findUser;
        try {
            UsersEntityMF usersEntityMF = new UsersEntityMF();
            EntityManager entityManager = usersEntityMF.getEntityManager();
            findUser = entityManager.find(User.class, name);
            entityManager.close();
            usersEntityMF.close();
            System.out.println("Выполнено.");
        }catch (NullPointerException e){
            throw new RuntimeException("Такого пользователя не существует.");
        }
        return findUser;
    }
}
```

UserEntityMF

```
package com.lab1.database.dao;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
```

```
class UsersEntityMF {

   private EntityManagerFactory emFactory;

   EntityManager getEntityManager(){
      emFactory = Persistence.createEntityManagerFactory("UserPersistenceUnit");
      return emFactory.createEntityManager();
   }

   void close(){
      emFactory.close();
   }
}
```

Book

```
package com.lab1.database.model;

import javax.persistence.*;

@Entity
@Table(name = "books")
public class Book {

   @Id
   @GeneratedValue(generator = "increment")
   private int id;
   @Column(name = "name")
   private String name;
   @Column(name = "author")
   private String author;
   @Column(name = "year")
   private int year;
   @Column(name = "language")
   private String language;
   @Column(name="genre")
   private String genre;
   @Column(name="price")
   private double price;
   @Column(name="description")
   private String description;
   @Column(name="link")
   private String link;
   @Column(name="encryptedImage")
   private String encryptedImage;

   public Book() {
   }

   public Book(String name, String author, int year, String language, String genre, double price,
String description, String link, String encryptedImage) {
      this.name = name;
```

```
    this.author = author;
    this.year = year;
    this.language = language;
    this.genre = genre;
    this.price = price;
    this.description=description;
    this.link = link;
    this.encryptedImage = encryptedImage;
}

public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getAuthor() {
    return author;
}
public void setAuthor(String author) {
    this.author = author;
}
public int getYear() {
    return year;
}
public void setYear(int year) {
    this.year = year;
}
public String getLanguage() {
    return language;
}
public void setLanguage(String language) {
    this.language = language;
}
public String getGenre() {
    return genre;
}
public void setGenre(String genre) {
    this.genre = genre;
}
public double getPrice() {
    return price;
}
public void setPrice(double price) {
    this.price = price;
}
public String getDescription() {
    return description;
```

```
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getLink() {
        return link;
    }
    public void setLink(String downloadLink) {
        this.link = downloadLink;
    }
    public String getEncryptedImage() {
        return encryptedImage;
    }
    public void setEncryptedImage(String encryptedImage) {
        this.encryptedImage = encryptedImage;
    }

    @Override
    public String toString() {
        return getClass().getSimpleName()+" ["+id+" "+name+" "+author+" "+year+"
"+language+" "+genre+" "+price+" "+description+"]";
    }
}
```

User

```
package com.lab1.database.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "users")
public class User {

    @Id
    @Column(name = "username")
    private String username;
    @Column(name = "password")
    private String password;
    @Column(name = "y_e")
    private double y_e;

    public User() {
    }

    public User(String username, String password, double y_e) {
        this.username = username;
        this.password = password;
        this.y_e = y_e;
```

```
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public double getY_e() {
        return y_e;
    }
    public void setY_e(double y_e) {
        this.y_e = y_e;
    }

    @Override
    public String toString() {
        return getClass().getSimpleName()+"["+username+"]";
    }
}
```

UserRole

```
package com.lab1.database.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "users_roles")
public class UserRole {

    @Id
    @Column(name = "username")
    private String username;
    @Column(name = "rolename")
    private String rolename;

    public UserRole() {
    }

    public UserRole(String username, String rolename) {
        this.username = username;
        this.rolename = rolename;
```

```
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getRolename() {
        return rolename;
    }
    public void setRolename(String rolename) {
        this.rolename = rolename;
    }

    @Override
    public String toString() {
        return getClass().getSimpleName()+"[username: "+username+" role: "+rolename+"]\n";

    }
}
```

BookService

```
package com.lab1.database.service;

import com.lab1.database.dao.BookDAO;
import com.lab1.database.model.Book;
import com.lab1.database.model.Format;

import java.io.File;
import java.util.List;
import java.util.Objects;

public class BookService {

    private BookDAO bookDAO = new BookDAO();

    public List<Book> getAllData() {
        return bookDAO.getAllBooks();
    }

    public void findData(int id) {
        bookDAO.findData(id);
    }

    public void addData(Book book) {
        bookDAO.addData(book);
    }

    public void updateData(Book book, int id) {
        bookDAO.updateData(book,id);
```

```
    }

    public void deleteData(int id,String name) {

        String pathname="d:/uploads/";
        bookDAO.deleteData(id);

        for(Format format : Format.values()) {
            if (new File(pathname+name + format.getFormat()).delete()) {
                System.out.println("Все файлы с именем "+name+" удалены");
            }
        }
    }

    public String findDataById(int id) {
        return bookDAO.findDataById(id);
    }

    public Book findBookById(int id) {
        return bookDAO.findBookById(id);
    }

    public boolean bookIsExist(String bookName){
        for (Book book:getAllData()){
            if (bookName.equals(book.getName())){
                return true;
            }
        }
        return false;
    }
}
```

UserService

```
package com.lab1.database.service;

import com.lab1.database.dao.UserDAO;
import com.lab1.database.model.UserRole;
import com.lab1.database.model.User;

import java.util.List;

public class UserService {

    private UserDAO userDAO=new UserDAO();

    public List<UserRole> getAllUserRole() {
        return userDAO.getAllUserRole();
    }

    public void add(User newUser, String role) {
        userDAO.add(newUser,role);
```

```
    }

    public void deleteUser(String username) {
        userDAO.delete(username);
    }

    public User findUserByName(String name) {
        return userDAO.findUserByName(name);
    }

    public void updateData(User user, String name) {
        userDAO.updateData(user,name);
    }

    public boolean isExist(String username){

        boolean isExist=false;

        for(UserRole userRole:getAllUserRole()){
            if(userRole.getUsername().equals(username)) {
                isExist = true;
            }
        }
        return isExist;
    }
}
```

AdminController

```
package com.lab1.database.controller;

import com.lab1.database.model.User;
import com.lab1.database.service.UserService;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.util.List;

@ManagedBean
public class AdminController {

    //Для формы добавления пользователя в БД
    private String username;
    private String password;
    private String confirmPassword;

    private List users;
    private UserService userService=new UserService();

    public List getUsers(){
        users=userService.getAllUserRole();
```

```
        return users;
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getConfirmPassword() {
        return confirmPassword;
    }
    public void setConfirmPassword(String confirmPassword) {
        this.confirmPassword = confirmPassword;
    }
    public void deleteUser(String name){
        userService.deleteUser(name);
    }

    public String getCurrentUserName() {
        HttpServletRequest request = (HttpServletRequest)
FacesContext.getCurrentInstance().getExternalContext().getRequest();
        return  request.getRemoteUser();
    }

    public void addUser() throws IOException {

        User newUser = new User(username,password,0.0);

        if (getUsername().equals("")||getPassword().equals("")) {
            FacesContext context = FacesContext.getCurrentInstance();
            context.getExternalContext().redirect("http://localhost:8080/addErrorServlet");
        }
        else {
            if (!userService.isExist(getUsername())) {
                if (password.equals(confirmPassword)) {
                    userService.add(newUser, "user");
                } else {
                    FacesContext context = FacesContext.getCurrentInstance();

context.getExternalContext().redirect("http://localhost:8080/addUserPasswordErrorServlet");
                }
            }
            else {
                FacesContext context = FacesContext.getCurrentInstance();
                context.getExternalContext().redirect("http://localhost:8080/addUserErrorServlet");
            }
        }
```

```
    }
}
```

ManagerController

```
package com.lab1.database.controller;

import com.lab1.database.model.Book;
import com.lab1.database.service.BookService;

import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ComponentSystemEvent;
import java.io.IOException;
import java.io.Serializable;
import java.util.List;

@ManagedBean
@ViewScoped
public class ManagerController implements Serializable {

    private List<Book> books;
    private List<Book> filteredBooks;
    private List<Book> sortedBooks;
    private BookService bookService = new BookService();

    public List<Book> getSortedBooks(){
        return books;
    }
    public List<Book> getFilteredBooks() {
        return filteredBooks;
    }
    public void setFilteredBooks(List<Book> filteredBooks) {
        this.filteredBooks = filteredBooks;
    }
    public void setSortedBooks(List<Book> sortedBooks) {
        this.sortedBooks = sortedBooks;
    }
    public List<Book> getBooks(){
        books=bookService.getAllData();
        return books;
    }

    public void deleteBook(int id,String name) throws IOException {
        FacesContext context = FacesContext.getCurrentInstance();
        bookService.deleteData(id,name);

context.getExternalContext().redirect("http://localhost:8080/manager_pages/manager_hello.xhtml");
    }
```

```
    public void beforeLoadingPage(){
        if (!FacesContext.getCurrentInstance().isPostback()){
            getBooks();
        }
    }
}
```

UserController

```
package com.lab1.database.controller;

import com.lab1.database.model.Book;
import com.lab1.database.model.Format;
import com.lab1.database.service.BookService;
import com.lab1.database.model.User;
import com.lab1.database.service.UserService;
import com.spire.pdf.FileFormat;
import com.spire.pdf.PdfDocument;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import java.util.Objects;

@ManagedBean
public class UserController {

    private double y_e;
    private int bookId;
    private UserService userService=new UserService();
    private BookService bookService=new BookService();
    private FacesContext context = FacesContext.getCurrentInstance();
    private HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
    private HttpServletRequest request = (HttpServletRequest)
FacesContext.getCurrentInstance().getExternalContext().getRequest();
    private String userName=request.getRemoteUser(); // Получаем имя текущего пользователя
    private User user=userService.findUserByName(userName); //Получаем пользователя по
имени
    private String filePath="d:/uploads/";

    public double getY_e() {
        return y_e;
    }
    public void setY_e(double y_e) {
        this.y_e = y_e;
    }
    public int getBookId() {
```

```
        return bookId;
    }
    public void setBookId(int bookId) {
        this.bookId = bookId;
    }

    public void refill(){

        User user = userService.findUserByName(userName);
        user.setY_e(user.getY_e()+getY_e());
        userService.updateData(user,user.getUsername());
    }

    public double getBalance(){

        AdminController adminController = new AdminController();
        String userName = adminController.getCurrentUserName();
        return userService.findUserByName(userName).getY_e();
    }

    public void checkForDownloadWithValue(String to,String value) throws IOException {

        Book book = bookService.findBookById(getBookId()); //Получаем книгу по id

        if(book == null) { //Проверяем есть ли книга по такому id
            response.sendRedirect(request.getContextPath() +
"/user_pages/searchBookError.xhtml");
        } else {
            String PDFFile = book.getLink() + Format.PDF.getFormat();

            if (user.getY_e() - book.getPrice() < 0) {//Проверяем есть ли деньги у пользователя
                response.sendError(HttpServletResponse.SC_BAD_REQUEST, "You need to top up
your balance.");
            } else {

                user.setY_e(user.getY_e() - book.getPrice());
                userService.updateData(user, user.getUsername());

                File file = new File(PDFFile);

                if (!file.exists()) {

                    user.setY_e(user.getY_e() + book.getPrice());
                    userService.updateData(user, user.getUsername());
                    response.sendError(HttpServletResponse.SC_NOT_FOUND);
                } else {
                    if (to.equals("convertPDFtoTXT")){
                        convertPDFtoTXT(file);
                    } else if(to.equals("convertPDFtoDOC")){
                        convertPDFtoDOC(file);
                    }else {
                        downloadWithValue(file,value);
                    }
                }
```

```
        }
      }
    }

    private void convertPDFtoTXT(File file) throws IOException {

        File txtFile = new File( filePath+file.getName().substring(0,file.getName().indexOf(".")) +
Format.TXT.getFormat());

        if (!searchConvertedFile(filePath, txtFile)) {

            PDDocument newPdDocument = PDDocument.load(file);

            if (!newPdDocument.isEncrypted()) {

                PDFTextStripper pdfTextStripper = new PDFTextStripper();
                String text = pdfTextStripper.getText(newPdDocument);
                FileWriter fileWriter = new FileWriter(txtFile);
                fileWriter.write(text);
                fileWriter.close();
            }
            newPdDocument.close();
        }
        downloadWithValue(txtFile,"attachment");
    }

    private void convertPDFtoDOC(File file) throws IOException {

        String link;

        link=filePath+file.getName().substring(0,file.getName().indexOf(".")) +
Format.DOC.getFormat();

        File docFile = new File(link);

        if (!searchConvertedFile(filePath, docFile)) {
            PdfDocument docDocument = new PdfDocument();
            docDocument.loadFromFile(filePath+file.getName());
            docDocument.saveToFile(link, FileFormat.DOC);
            docDocument.close();
        }
        downloadWithValue(docFile,"attachment");
    }

    private void downloadWithValue(File file, String value) throws IOException {

        int readingData;
        final int myAppBuffSize = 209715200;//200 MB
        byte[] creatingBuffer;
        BufferedInputStream bufferedInputStream = null;
        BufferedOutputStream bufferedOutputStream = null;

        response.reset();
        response.setContentType("application/pdf");
```

```
        response.setContentLength((int) file.length());
        response.setHeader("Content-disposition", value+"; filename=" + file.getName());

        try{
            bufferedInputStream = new BufferedInputStream(new FileInputStream(file),
myAppBuffSize);
            bufferedOutputStream = new BufferedOutputStream(response.getOutputStream(),
myAppBuffSize);
            creatingBuffer = new byte[myAppBuffSize];

            while ((readingData = bufferedInputStream.read(creatingBuffer)) > 0) {
                bufferedOutputStream.write(creatingBuffer, 0, readingData);
            }

            bufferedOutputStream.flush();

        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            Objects.requireNonNull(bufferedInputStream).close();
            Objects.requireNonNull(bufferedOutputStream).close();
        }
        context.responseComplete();
    }

    private boolean searchConvertedFile(String filePath, File file){
        for (File searchConvertTXTFile : Objects.requireNonNull(new File(filePath).listFiles())){
            if (searchConvertTXTFile.equals(file)){
                return true;
            }
        }
        return false;
    }

}
```

LogoutServlet

```
package com.lab1.database.servlets;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/logoutServlet")
public class LogoutServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        request.getSession().invalidate();
```

```
        response.sendRedirect(request.getContextPath() + "/user_pages/user_hello.xhtml");
    }

}
```